sRNARFTarget: A machine learning-based approach for sRNA Target Prediction

by

A thesis submitted to the

School of Graduate Studies

in partial fulfillment of the

requirements for the degree of

Master of Science

Supervisor: Lourdes Peña-Castillo

Department of Computer Science

Memorial University of Newfoundland

July 2020

St. John's Newfoundland

Abstract

Bacterial small regulatory RNAs (sRNAs) play a vital role in the regulation of gene expression in bacteria. sRNAs regulate gene expression by interacting with mRNAs or proteins. Bacterial sRNAs are involved in various processes, such as environmental stress response, metabolism, and virulence. We need to identify the mRNAs and/or proteins that these sRNAs interact with, to understand the functional roles of sRNAs. These mRNAs or proteins are called targets of the sRNAs. There are several computational tools available for sRNA target prediction; however, these tools have a high number of false positives, and the most accurate tool requires sRNA sequence conservation across bacteria. As a result of this research project, a machine-learning-based method (sRNARFTarget) for sRNA target prediction applicable to any bacterium or sRNA has been developed. In this thesis, we show that sRNARFTarget substantially outperforms current non-comparative genomics-based methods in terms of running time and ranking of true interactions.

Contents

List of Figures

List of Tables

List of Abbreviations

AUPRC Area under the PR Curve

AUROC Area under the ROC Curve

GB Gradient boosting

KNN K-Nearest Neighbor

MDA Mean decrease in accuracy

RF Random Forest

Tetra nt. Tetranucleotide frequency

Tetra nt. Diff Difference between tetranucleotide frequency of sRNA and mRNA

Tetra nt. Diff & Dist. Difference between tetranucleotide frequency and distance between secondary structures of sRNA and mRNA

Tri nt. Trinucleotide frequency

Tri nt. Diff Difference between trinucleotide frequency of sRNA and mRNA

Tri nt. Diff & Dist. Difference between trinucleotide frequency and distance between secondary structures of sRNA and mRNA

Chapter 1

Introduction

Bioinformatics is the application of computational approaches to understand biological phenomena. With the increased generation of biological data, the opportunity of using machine learning models on biological data has emerged. In this thesis, we focus on an important question in the area of bacterial gene regulation. sRNAs are bacterial small regulatory RNAs, usually less than 200 nucleotides in length. sRNAs are also called non-coding RNAs as they are not translated into a protein. sRNAs play an essential role in gene expression regulation in bacteria and have become a rising class of regulatory RNAs [2]. They are involved in several biological functions such as virulence, metabolism, and environmental stress response [2]. The sRNAs exert their functions when they interact with mRNAs (messenger RNAs) or proteins. These mRNAs or proteins are called the targets of the sRNAs. There have been many sRNAs discovered in recent years; however, their corresponding targets are yet to be

found. To understand the roles and functions of sRNAs, it is important to find out their targets and thus, identifying targets of sRNAs has become an essential piece of bacterial RNA science.

There are several programs developed in previous studies for finding sRNA targets [1], such as CopraRNA [3], IntaRNA [4] and SPOT [5]. We will discuss more on these in Chapter 2. These programs generate many false positives which reduce the accuracy of the program. Here we developed a machine-learning based method to predict sRNA targets trained with data generated by RNA-seq based methods. Our method can be applied to any sRNA-mRNA pair (i.e., does not require sequence conservation of either sRNA or mRNA). We compare the performance of our method with that of CopraRNA and IntaRNA. Additionally, we implemented a pipeline to obtain the interpretations of predictions generated by sRNARFTarget program using existing interpretability programs.

This thesis is organized as follows: we will discuss related work about sRNA target prediction and interpretability of machine learning models in Chapter 2. Chapter 3 describes the methodology: data collection and processing, feature extraction, machine learning models' training, model selection, and benchmarking. Lastly, we will discuss the programs created for sRNARFTargets' interpretation. Chapter 4 presents results and discussion. Chapter 5 is the conclusion. The program code and supplementary files are available at the below link.

https://github.com/BioinformaticsLabAtMUN/sRNARFTarget.

Chapter 2

Related Work

2.1 sRNA Target Prediction

There have been many programs developed for sRNA target prediction in previous years. Some of these programs are CopraRNA [3], SPOT [5], TargetRNA2 [6], IntaRNA [4], TargetRNA [7], and RNAcofold [8]. The existing programs can be categorized into programs such as IntaRNA [4], RNAplex [9], CopraRNA [3], TargetRNA2 [6] and RNAup [10] that predict the actual targets and finds the most likely local interaction; and programs such as Pairfold [11], RNAcofold [8], RNAhybrid [12] and RNA duplex [13] that find the full interaction between the sRNA and the longer target RNA. Another categorization of RNA target prediction programs is programs for general RNA- RNA interactions and programs for sRNA-target predictions. Some of the existing sRNA-mRNA interaction programs are sTarpicker [14], TargetRNA [7],

RNApredator [15], CopraRNA, and SPOT.

The first prediction model for sRNA-mRNA interaction was presented by [16]. This model is based on the Smith-Waterman local sequence alignment interaction algorithm [17]. This model was designed for *Escherichia coli* and could not be applied to other bacteria. The second model for prediction is named TargetRNA [7]. In TargetRNA, the interaction between a given sRNA and a candidate mRNA target is predicted by calculating a hybridization score for the two RNA sequences. The hybridization score is calculated by an extension of the Smith-Waterman dynamic program [17]. TargetRNA does not account for the structures of either the sRNA or mRNA. TargetRNA2 [6] is an sRNA prediction web server. This program allows RNA-seq data to be incorporated. It uses several different features for prediction, such as conserved regions and secondary structures. Mandin et al. [18] proposed a model for sRNA target prediction by searching strong sRNA-mRNA duplexes. Each sRNA-mRNA duplex was scored as a sum of both positive and negative contributions, which correspond to pairing nucleotides and internal loops. The statistical significance of the duplex was used as the criterion for interaction.

IntaRNA is a general prediction program for RNA–RNA interaction. This program includes target site approachability, and users can define seeds. Interactions can be predicted for a single organism in IntaRNA. IntaRNA and RNAup have a similar performance on predicting sRNA targets and perform best among all studied programs, namely TargetRNA, RNAhybrid, and RNAplex. [19] introduced IntaRNA

2.0. It is an open-source reimplementation of the former approach IntaRNA. It facilitates improved and customizable RNA–RNA interaction prediction. The following features were incorporated in this approach, Seed stability constraints, and Dangling end contributions. sTarPicker [14] is a two-step model based on the hybridization between the interaction. In the first step, seed regions of sRNA and its target are bound with base-pairing. In the second step, the initial hybrid extends to create the entire sRNA-target interaction. RNApredator is a webserver for the prediction of sRNA targets. RNApredator predicts sRNA targets using RNAplex. RNApredator also considers the accessibility of the target to improve the prediction specificity. To enable fast computation, accessibility profiles are pre-computed using RNAplfold [20]. The result shows that the prediction accuracy of RNApredator is comparable to that of other methods like RNAup and IntaRNA. RNAcofold [8] is an RNA-RNA interaction program. It computes the base-pairing pattern, hybridization energy of interacting RNA pairs and calculates the minimum energy structure.

Currently, CopraRNA and SPOT are the most recent programs for sRNA target prediction. CopraRNA extends the functionality of IntaRNA and is based on comparative genomics [21]. The main feature of comparative genomics is that it looks for similar or conserved sequences among various bacteria. CopraRNA comes with certain limitations. One of the limitations is that it can only be applied to sRNAs whose sequence is conserved among different bacteria. Many sRNAs are specific to a single bacterium, and thus CopraRNA is unable to predict the mRNA targets of

these sRNAs.

SPOT stands for sRNA Target Prediction Organizing Tool. It is a computational software pipeline for sRNA-target prediction. It uses existing sRNA target prediction programs such as TargetRNA2, sTarPicker, IntaRNA and, CopraRNA to predict sRNA-mRNA interactions. These programs are run parallelly to search for interactions and results are collated for each program. Lastly, experimental data is integrated using customizable result filters. SPOT results show that it was able to find additional putative mRNA targets of sRNA RydC in addition to the already known target. SPOT sensitivity was equal to or surpassed any specific method when applied on 12 characterized sRNAs.

Backofen, et al.[22] examines the merits and demerits of the existing programs and tools for sRNA discovery and their target predictions. There have been two comparative assessments of RNA - RNA interaction prediction [[1], [23]]. Pain et al.'s results show that CopraRNA outperformed all other programs in terms of accuracy.

Figure 2.1 shows the results obtained by [1]. RNAplex, IntaRNA, RNAhybrid, RNAup, RNAcofold, RNAduplex, and Pairfold were run using the Unix command-line versions of the programs and, TargetRNA2 and CopraRNA were run from their webservers. The results show that CopraRNA performs better compared to other programs by predicting more number of trusted pairs. Results show that programs predicting local interactions, such as CopraRNA, IntaRNA, RNAplex, RNAup and TargetRNA2, outperform programs to predict complete RNA-RNA hybrid. Programs

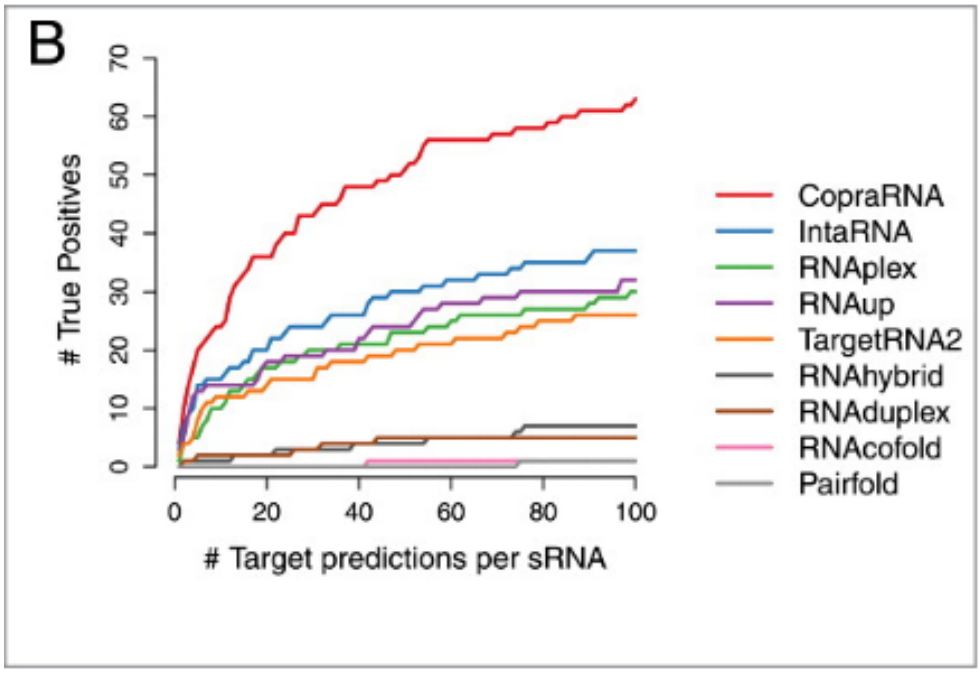


Figure 2.1: The ROC-like curve for each program in the assessment. Y-axis shows the number of trusted interactions predicted by each program, and the x-axis indicates the number of predictions with the best ranking. Each curve shows the number of trusted interactions predicted by the program among the predictions with the best ranking (Figure reproduced from [1] under CC-by-nc license).

that achieve the lowest performance are RNAcofold and Pairfold. Table 2.1 provides a summary of the sRNA target prediction programs discussed above.

Program	**Strategy**	**Limitations**
CopraRNA [3]	Based on comparative-genomics that requires sequence conservation of sRNA and mRNA.	Requires homologs of sRNA and mRNA in at least four bacteria. Runs one sRNA at a time.
IntaRNA [4]	Uses interaction sites accessibility and user-specified seed.	Long exceution time.
SPOT [5]	Metamethod combining four current tools - TargetRNA2, sTarPicker, IntaRNA and CopraRNA.	Runs one sRNA at a time. Requires Payment when run through AWS.
TargetRNA2 [6]	Uses four features: conservation of the sRNA, sRNA accessibility, mRNA accessibility and hybridization energy.	Requires conservation of the sRNA in other bacteria.
RNAcofold [8]	Computes base-pairing pattern and hybridization energy.	Neglects some important interaction structures and is limited to dimeric complexes.
StarPicker [14]	Two-step model based on the hybridization between the sRNA-mRNA interaction.	No longer available.
RNApredator [15]	Based on RNAplex. Uses RNAplfold to precompute the accessibility profiles for all genomes.	Accuracy is similar to that of other methods such as RNAup and IntaRNA.

Table 2.1: Summary of the sRNA target prediction programs

Most of the current programs for sRNA target predictions use by default untranslated regions (UTR) [24], which is a part of the whole sequence, not the whole sequence. UTR refers to both sides of a coding sequence on an mRNA strand, one on either side. It is called 5' UTR when it is found on 5' side. Similarly, when it is located on 3' side, it is called 3' UTR.

There are wet-lab approaches that use RNA sequencing (RNA-seq) [25] to identify targets of sRNAs. Some of these approaches are MAPS [26], GRIL-seq [27], CLASH [28] and, RIL-seq [29]. MS2 affinity purification coupled with RNA-seq (MAPS) method identifies RNAs (mRNA, tRNA, or sRNA) that interact with specific individual sRNA. In this method, RNA targets were co-purified by fusing MS2 tag to *E. coli* sRNAs. Combined with RNA-seq, MS2-sRNA affinity purification uncovers the targets of a specific sRNA. GRIL-seq stands for Global small non-coding RNA target identification by ligation and sequencing. This method also identifies targets for a specific sRNA. The approach exploits the advantage of the sRNA and its target mRNA's proximity stabilized by Hfq protein. GRIL-seq identified direct regulatory targets of sRNA *prrF1* in *Pseudomonas aeruginosa*. CLASH [28] retrieves base-paired sRNA–mRNA duplexes using UV-crosslinking, ligation and sequencing of hybrids. This method identified several mRNA targets for sRNA *esr41* of *E. coli*. RIL-seq (RNA interaction by ligation and sequencing) [29] is an experimental-computational approach. It detects Hfq-bound sRNA-target pairs in bacteria. These methods have significantly increased the number of known sRNA-mRNA interactions.

2.2 Interpretability

Many machine learning models remain black boxes [30] despite their high predictive performance, as it is hard to comprehend the role of the features when making predictions. Interpretability is understanding the reasoning behind the outcome of the model. Model interpretability is essential as it shows the features that impact the model's outcome and allows a user to explain how the model arrived at its predictions. Interpretability comes after the model has made its predictions [31].

Interpretability methods can be categorized in two ways. Model agnostic or model specific. Model agnostic methods can be applied to any machine learning model, unlike model-specific methods that could be applied to specific kinds of machine learning models. It can further be categorized in terms of scope, local or global. Global methods give an understanding of how the model makes its prediction based on its features and model structure. Local interpretations provide an understanding of what features are influencing the predicted outcome for a given observation [31].

There are multiple surveys on machine learning interpretability such as [32], [33], [34]. [35] used Shapley Additive explanations (SHAP) [36] to explain the predictions of XGBoost model that classifies patients into four laser surgery categories and states that the explanations generated by SHAP for the results were in line with antecedent knowledge from specialists. [37] carried out the implementation of the Random forest model to predict ICU mortality for precision medicine data and added interpretability to the model's output by using LIME [30]. The result shows that the simple

generated explanations from a complex model were consistent with current medical understanding and was able to interpret the influence of the features on prediction. We considered two python packages for sRNARFTarget interpretability; SHAP [36] [38] and pyCeterisParibus [39].

SHAP is a model agnostic approach and can be used for local and global interpretations. Shapley value is the average marginal contribution of a feature value across all possible coalitions (different combination sets of features). The Shapley value is a process that, in terms of game theory, assigns the payouts to players according to their contribution to the total payout. In machine learning, the prediction task is analogous to the game, the model outcome is the payout, and the features are the players so that the Shapley value estimates the contribution of each feature towards the final prediction [40].

pyCeterisParibus is a model agnostic approach and can be used for instance based (local) interpretations. It is based on Ceteris paribus profiles of R [41], Ceteris paribus profiles are individual variable profiles generated by changing the value of one feature at one time and keeping all other feature values constant. They are also called what-if profiles.

2.3 Summary

There are many programs for sRNA target prediction. Among those, CopraRNA is the most accurate one; however, it requires sequence conservation of the sRNAs and

mRNAs in at least four bacterial species. Out of the programs that are not comparative genomic-based as CopraRNA, IntaRNA and sTarPicker have been shown to achieve the best results in terms of the area under the ROC curve (AUROC). Recently, several RNA sequencing based methods have been developed to experimentally identify targets of sRNAs. These methods have increased the number of known sRNA-mRNA interactions.

Chapter 3

Methodology

The developed sRNA target prediction program sRNARFTarget is generated using the Random Forest [42] machine learning algorithm. The idea behind this approach is that the machine learning classifier receives instances containing the difference between trinucleotide frequency of sRNA and mRNA pairs. The random forest classifier then returns the prediction probability of the interaction of these two RNAs.

Figure 3.1 gives an overview of the proposed approach. Sequence derived features (Trinucleotide frequency difference) are extracted from the sequences of sRNA - mRNA pairs. These observations are given to a machine learning-based method, which then predicts whether the two RNAs interact or not.

To train the Random Forest, we first collected experimentally determined sRNA-mRNA pairs from the literature (Section 3.1), extracted features from their sequences such as k-mer frequency and secondary structure distances (Section 3.2), carried out

a grid search cross-validation to select the best performing model in terms of AUROC (Section 3.3), and finally comparatively assessed the performance of the final model with two state-of-the-art programs (CopraRNA and IntaRNA) on three independent data sets (Section 3.5).

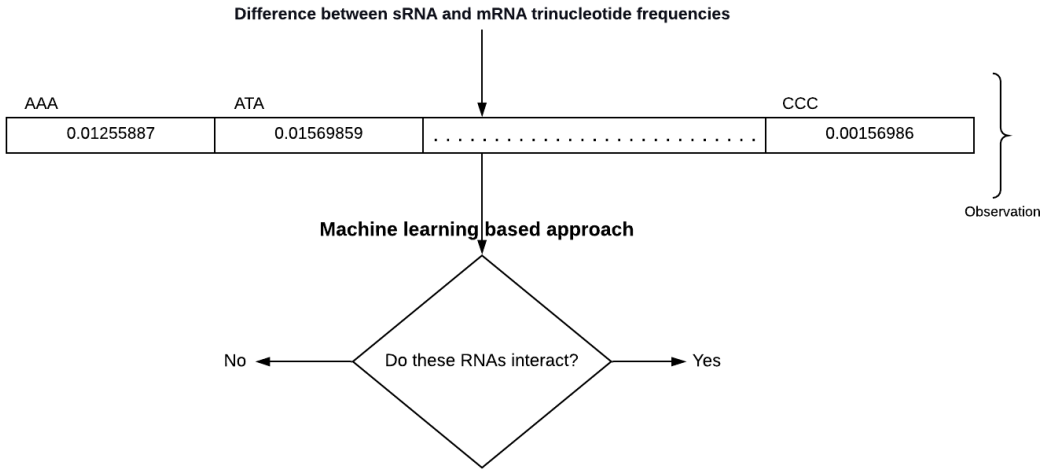


Figure 3.1: Workflow of Machine Learning based approach

3.1 Data Collection

By searching in NCBI Pubmed, we identified studies where sRNA-mRNA interactions were identified (see Table 3.1). We collected all studies found that provided sRNA - mRNA validated pairs and then collected all sRNA-mRNA pairs listed in these studies. We gathered roughly 2400 pairs from multiple bacteria.

Bacteria Name	**Reference**
Escherichia coli	[1], [43], [28], [44], [45]
Pseudomonas aeruginosa	[46], [27], [47]
Burkholderia cepacia	[47]
Pasteurella multocida	[48]
Salmonella	[49], [50]
Mycobacterium tuberculosis	[51]
Synechocystis	[52] [53]
Multiple bacteria	[54], [55], [56]

Table 3.1: Studies identifiying sRNA targets in the literature

3.2 Data processing

The data available in the literature are in different formats. Few datasets were available with sRNA - mRNA names, accession numbers, sRNA - mRNA sequences or, locations of sRNA - mRNA in the genome. We used the given sequences directly if they were provided such as from sTarBase3.0 [55]. For other datasets, we created a data file which contains Entrez genome accession number, sRNA and target mRNA name.

3.2.1 Split data into training and benchmarking set

We divided the collected data into training data and benchmarking data.

Benchmarking data: We kept the following from the collected dataset to use later for benchmarking. 102, 22, and 20 sRNA-mRNA pairs from *Escherichia coli* [1], *Pasteurella multocida* [48] and *Synechocystis* [52] [53] respectively. Table 3.2 shows genome accession number, number of sRNAs per strain, number of pairs per strain used in this project and a total of all columns at the end of the Table.

Accession #	**Strain**	**No. of sRNAs**	**No. of pairs**
NC_000913.3	*Escherichia coli str. K-12 substr. MG1655*	22	102
NC_002663.1	*Pasteurella multocida subsp. multocida str. Pm70*	1	22
NC_000911.1	*Synechocystis sp. PCC 6803*	2	20
Total	***3***	***25***	***144***

Table 3.2: Benchmarking Data

Training data: The remaining data was used for training the models. Table 3.3 shows the genome accession number, number of sRNAs per strain, the number of pairs per strain used in this project and a total count of all columns at the end of the Table.

Accession #	**Strain**	**No. of sRNAs**	**No. of pairs**
NC_003062.2	*Agrobacterium fabrum str. C58*	1	9
NC_011312.1	*Aliivibrio salmonicida LFI1238*	1	1
NC_012560.1	*Azotobacter vinelandii DJ*	1	10
NC_000964.3	*Bacillus subtilis subsp. subtilis str. 168*	1	12
NC_007618.1	*Brucella abortus 2308*	6	10
NC_011000.1	*Burkholderia cenocepacia J2315*	1	1
NC_011916.1	*Caulobacter crescentus NA1000*	1	1
NC_003366.1	*Clostridium perfringens str. 13*	1	1
NC_002695.1	*Escherichia coli O157:H7 str. Sakai*	7	28
NC_000913.3	*Escherichia coli str. K-12 substr. MG1655*	42	358
NC_007880.1	*Francisella tularensis subsp. holarctica LVS*	1	1
NC_000915.1	*Helicobacter pylori 26695*	2	2
NC_003210.1	*Listeria monocytogenes EGD-e*	2	5
NC_018588.1	*Listeria monocytogenes serotype*	5	5
NC_000962.3	*Mycobacterium tuberculosis H37Rv*	1	9
NC_002946.2	*Neisseria gonorrhoeae FA 1090*	1	4
NC_003112.2	*Neisseria meningitidis MC58*	2	5

Table 3.3 continued from previous page

NC_005072.1	*Prochlorococcus marinus subsp. pastoris str. CCMP1986*	1	4
NC_002516.2	*Pseudomonas aeruginosa PAO1*	7	78
NC_007493.2	*Rhodobacter sphaeroides 2.4.1*	5	6
NC_003198.1	*Salmonella enterica subsp. enterica serovar Typhi str. CT18*	1	1
NC_016856.1	*Salmonella enterica subsp. enterica serovar Typhimurium str. 14028S*	1	11
NC_003197.1	*Salmonella enterica subsp. enterica serovar Typhimurium str. LT2*	40	64
NC_016810.1	*Salmonella enterica subsp. enterica serovar Typhimurium str. SL1344*	8	41
NC_003047.1	*Sinorhizobium meliloti 1021*	1	1
NC_022222.1	*Staphylococcus aureus subsp. aureus 6850*	2	8
NC_007795.1	*Staphylococcus aureus subsp. aureus NCTC 8325*	2	2
NC_002745.2	*Staphylococcus aureus subsp. aureus N315*	2	5
NC_008022.1	*Streptococcus pyogenes MGAS10270*	1	1
NC_003888.3	*Streptomyces coelicolor A3(2)*	1	1
NC_000911.1	*Synechocystis sp. PCC 6803*	1	1

Table 3.3 continued from previous page

NC_009783.1	*Vibrio campbellii ATCC BAA-1116*	5	11
NC_009784.1	*Vibrio campbellii ATCC BAA-1116*	3	3
NC_022270.1	*Vibrio campbellii ATCC BAA-1116*	1	2
NC_002505.1	*Vibrio cholerae O1 biovar El Tor str. N16961*	7	10
NC_002506.1	*Vibrio cholerae O1 biovar El Tor str. N16961*	6	28
NC_009457.1	*Vibrio cholerae O395*	1	1
NC_004603.1	*Vibrio parahaemolyticus RIMD 2210633*	1	1
NC_003131.1	*Yersinia pestis CO92*	1	1
NC_010465.1	*Yersinia pseudotuberculosis YPIII*	2	2
Total	***37***	***176***	***745***

Table 3.3: Model Training Data

3.2.2 Get training data

Our first data preprocessing step was to remove any duplicate pair. Next, we wrote Nextflow [57] pipelines to get the complete sRNA and mRNA sequences from NCBI, calculate the k-mer frequency and obtain the k-mer frequency difference. This pipeline is shown in Figure 3.2.

1. We run the training data through the first nextflow pipeline, Filtergenes. This pipeline finds whether the sRNAs and mRNAs exist in NCBI Gene database

using the esearch function of Entrez direct [58] and generates a final output containing sRNA-mRNA pairs found in NCBI.

2. Using output from the first pipeline, we run pipeline 2 to get the sRNA/mRNA sequences. We use esearch from Entrez direct, bedtools [59] biocontainer [60] in this pipeline.

3. We combine sequences retrieved from pipeline 2 and the ones which were directly collected [55]. We then run the pipeline 3 to extract k-mer frequency from sequences as shown in Figure 3.2. This pipeline uses skbio [61] module of python to extract k-mer frequency from sequences.

3.2.3 Secondary structure distance

We retrieve the distance between the sRNA and mRNA secondary structure from the sequences. This is achieved with the following steps.

1. Obtain predicted secondary structure using the CentroidFold [62] program to get the secondary structure of sRNAs and mRNAs. This program takes input sequences and returns the secondary structures of sequences.

2. Calculate the distance between sRNA and mRNA secondary structures using RNAdistance [63] program. This program takes input secondary structures of RNAs pairwise and calculates the distance based on the value passed in the

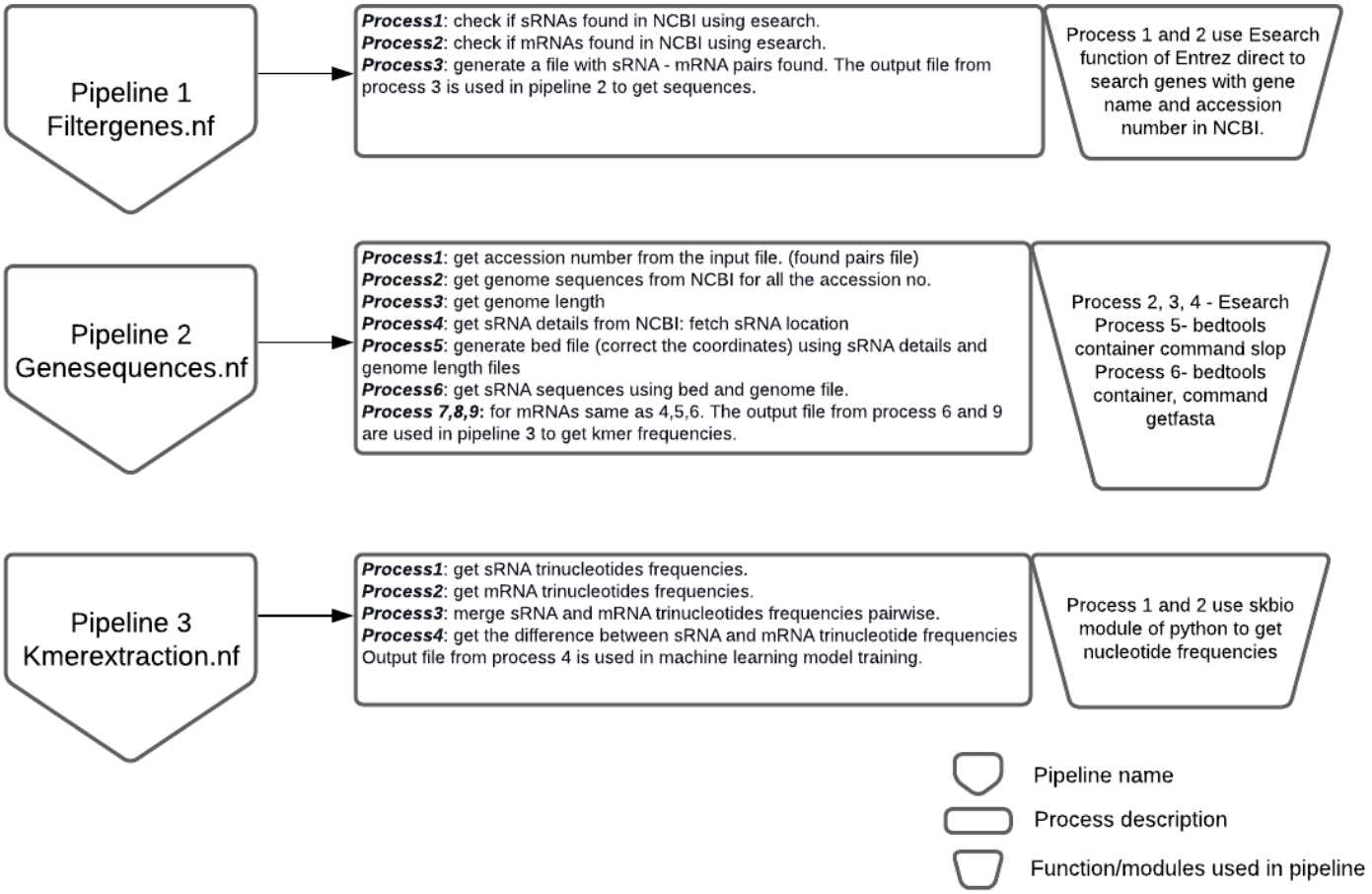


Figure 3.2: Nextflow pipelines for data processing and k-mer extraction

distance parameter. We calculated distance for all the values of the distance parameter.

3.3 Machine learning model selection

We generate models for sRNA target prediction using three ML methods, namely, Random Forest (RF), K-nearest neighbors (KNN) and gradient boosting (GB). We

used SKlearn [64] APIs of Random Forest, KNN and, Gradient Boosting machine learning algorithms to implement these classifiers.

3.3.1 Training data for binary classification

As a result of data processing, we had a final training dataset containing 745 true positive sRNA-mRNA pairs. We created 745 true negatives by randomly permuting the sRNA-mRNA pairs, and then processed the permuted pairs through the pipelines to calculate the k-mer frequency and secondary structure distance.

3.3.2 Model training

We used R importance function [65] based on mean decrease in accuracy to get the feature importance, and filter out any feature with a mean decrease in accuracy $\leq$ 0. We used Grid search cross-validation API of scikit-learn to get the best estimator/parameters for the models. See Table 3.4 for parameter grids used in Grid search CV. We did 10 fold stratified cross-validation to ensure balanced class distribution in each fold and avoid overfitting. We used the area under the ROC curve (AUROC) [66] and average precision [67] to evaluate the performance of the models.

Model	Parameters	Values
Random Forest	Number of trees (n_estimators)	[500, 600, 800, 1000]
	Number of features for split (max_features)	['sqrt', 'log2']
	Maximum depth of the tree (max_depth)	range(1, 11)
Gradient Boosting	n_estimators	[400,500,700,1000]
	max_features	["log2","sqrt"]
	max_depth	range(1, 11)
Kneighbors (KNN)	n_neighbors	range(1, 50)
	weights	['distance', 'uniform']

Table 3.4: Parameters for Grid search cross-validation

3.3.3 Model selection

See Table 3.5 for the count of observations and features for each data set. As a proof of concept and to be able to assess the effect of a larger dataset for training, we used two sets of features extracted from a small training data consisting of 102 *E. coli* pairs [1]: 1) Trinucleotide frequency of sRNA and mRNA sequences (128 features in total) 2) Tetranucleotide frequency of sRNA and mRNA sequences (512 features in total). We then used two more sets of features on the same data as above: 1) Trinucleotide frequency difference (64 features) 2) Tetranucleotide frequency difference (256 features). Discussion on why these features were selected can be seen in Chapter

4.

As mentioned above, models' performance was evaluated using 10-fold cross-validation. Based on the results obtained (discussed in Chapter 4), we decided to use as features the trinucleotide frequency difference and tetranucleotide frequency difference and generate new models training on the entire training data containing 1490 observations (745 true positives and 745 true negatives). For each set of features, we found the optimal parameter setting for each classifier using grid search CV, and compared the models' performance in terms of AUROC an average precision.

To explore whether other features will increase the performance of the best model (i.e., generated with trinucleotide frequency difference), we retrieved sRNA-mRNA secondary structure distances as discussed in section 3.2.2 and add them as extra features to the trinucleotide frequency difference for a total of 71 features (64 trinucleotide frequency difference and 7 distance features).

We got feature importance based on the mean decrease in accuracy from R function for entire training data with trinucleotide frequency difference, and all the features had a mean decrease in accuracy greater than zero. Feature importance plot can be seen in Chapter 4. We selected the model with the highest AUROC and average precision as our final model. We saved this model to be used by the nextflow pipeline implementing the sRNARFTarget program.

Data set	Feature name	True positives	True negatives	No. of features
Pilot study set	Trinucleotide frequency	102	102	128
	Tetranucleotide frequency	102	102	512
	Trinucleotide frequency difference	102	102	64
	Tetranucleotide frequency difference	102	102	256
Large training set	Trinucleotide frequency difference	745	745	64
	Tetranucleotide frequency difference	745	745	256
	Trinucleotide frequency difference with secondary structure distance	745	745	71
	Tetranucleotide frequency difference with secondary structure distance	745	745	263

Table 3.5: Count of features and instances in training datasets

3.4 sRNARFTarget nextflow pipeline

We created a nextflow pipeline that uses the saved random forest model for sRNA target prediction. The pipeline takes FASTA files as input: a fasta file with sRNA sequences and another fasta file with mRNA sequences. The final result of the pipeline is a CSV file containing prediction probabilities of sRNA-mRNA interaction sorted in descending order with the sRNA-mRNA ID. Next, we discuss each process of

sRNARFTarget pipeline. Figure 3.3 presents the workflow of sRNARFTarget program.

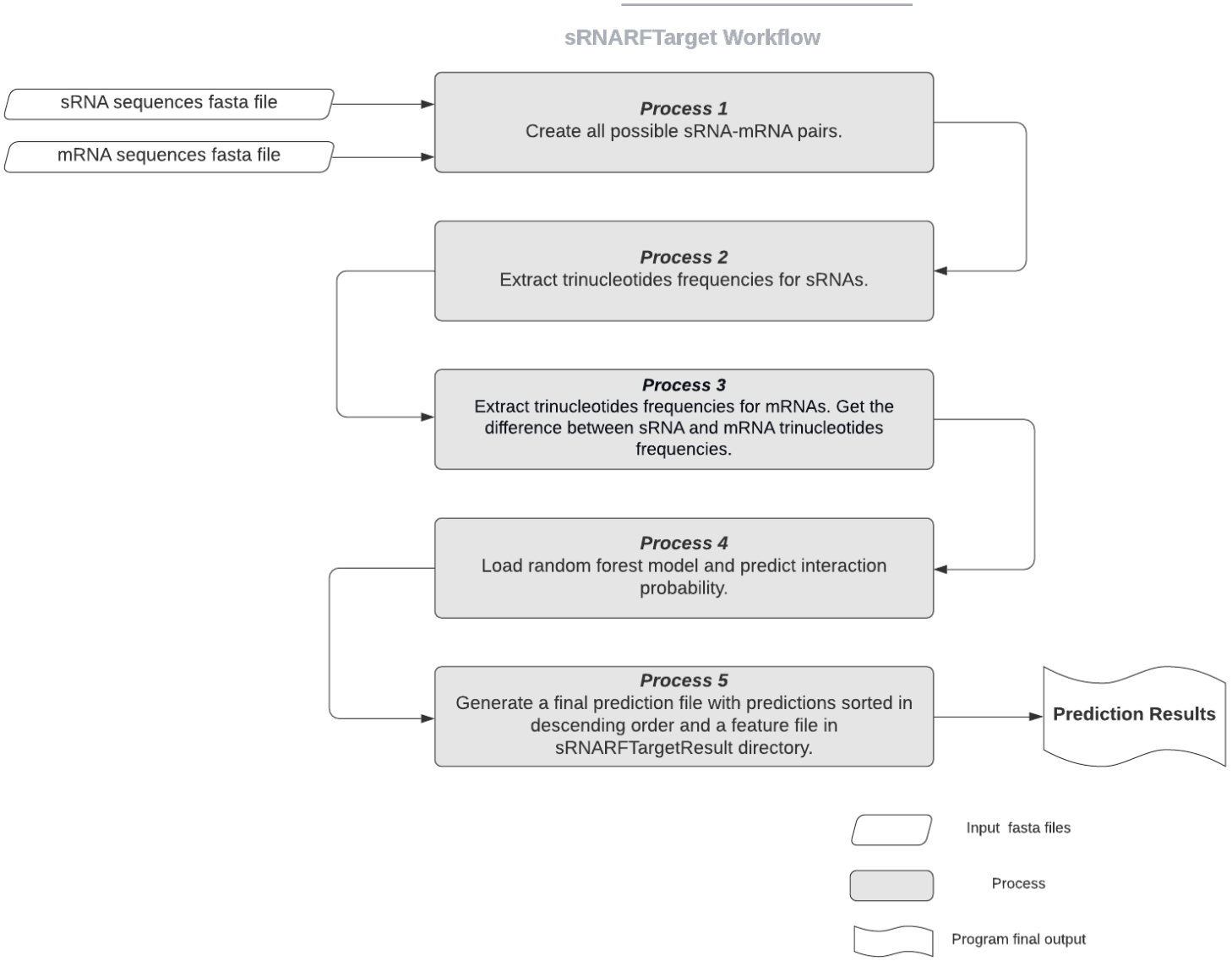


Figure 3.3: sRNARFTarget workflow

- The first process takes two input fasta files for sRNA and mRNA sequences. This process creates all possible pairs from the input sRNA and mRNA sequences. Each sRNA is paired with all mRNAs. For example, if the input sRNA file has 5 sRNA sequences and mRNA file has 9 mRNA sequences, then it will create 45 sRNA-mRNA pairs, 9 pairs for each sRNA.

- Process 2 extracts trinucleotide frequency for sRNA sequences of all possible pairs created in the first process. It uses the skbio [61] module of python to extract the k-mer frequency from sequences.

- Process 3 extracts trinucleotide frequency for mRNAs of all possible pairs created in the first process using the skbio python module. It then gets the difference between sRNA and mRNA trinucleotide frequency by subtracting sRNA frequency from mRNA frequency. It uses pandas [68] subtract function to do this.

- Process 4 receives sRNA-mRNA trinucleotide frequency difference from the last process. It loads the saved random forest model and makes predictions for all pairs. It generates the result file containing three columns sRNA ID, mRNA ID and predicted interaction probability.

- The last process creates a directory 'sRNARFTargetResult' and generates two files in it. First is the final prediction result file 'Prediction_probabilities.csv' containing the results sorted by predicted interaction probability from high to

low, rounded to five decimals. The second file is the 'FeatureFile.csv' that contains features for sRNA and mRNA pairs. It consists of sRNA/mRNA IDs and corresponding trinucleotide frequency difference. The feature file is later used by the interpretability programs.

3.5 Benchmarking

Based on comparative assessments of sRNA target prediction programs [1, 5], four programs (CopraRNA, IntaRNA, SPOT and sTarPicker) are reported to have the best performance with CopraRNA been the best performing program. SPOT is reported to be comparable to CopraRNA; however, we were unable to run SPOT locally and running SPOT through AWS [69] requires payment [70]. sTarPicker is no longer available. Therefore, we ran CopraRNA and IntaRNA for our benchmark. CopraRNA and IntaRNA are non - machine learning based programs.

3.5.1 Data for benchmarking

The data used for independent benchmarking (i.e., these data were not seen during training) have 22 sRNAs and 102 sRNA-mRNA pairs for *E. coli* [1] , 1 sRNA and 22 pairs for *P. multocida* [48], 2 sRNAs and 20 pairs for *Synechocystis* bacteria [52], [53]. For *E. coli*, we extracted the sequences for 22 sRNAs using the nextflow pipeline. For all other sRNAs, we fetched the sequence directly from the NCBI nucleotide database. The location of *isar1* sRNA was taken as reported in [52]. The location

of *psrR1* sRNA (1671919-1672052) was confirmed by oral communication with the author of [53]. Finally, *gcvB* sRNA location was obtained from [48].

As we wanted to perform a transcriptome-wide prediction, we collected location details for all the mRNAs belonging to each bacterium directly from NCBI. We then extracted the sequences for all the mRNAs per bacterium using the nextflow pipeline.

Input files for sRNARFTarget and IntaRNA consist of sRNA and mRNA sequences in each file and the count of sequences has been listed in Table 3.6.

To run CopraRNA, we used homologs provided in [52] and [53] for *isar1* and *psrR1* sRNAs of *Synechocystis* bacteria. For *gcvB* sRNA of *P. multocida*, we retrieved homolog sRNAs from NCBI. To find homologs for *E. coli* sRNAs, we used GLASSgo - sRNA Homolog Finder program [71].

3.5.2 Running programs

3.5.2.1 sRNARFTarget

sRNARFTarget can be run from the Linux command line. It takes two arguments, an sRNA and an mRNA fasta file. We ran sRNARFTarget for each bacterium with the number of sequences shown in Table 3.6. As per the workflow of sRNARFTarget presented in Figure 3.3, the first process of pipeline generated 93280 sRNA-mRNA pairs for *E. coli*, 1804 pairs for *P. multocida* and 6358 pairs for *Synechocystis* bacteria. The final result generated the CSV files comprising the sorted prediction probabilities rounded to five decimal places for all pairs per bacterium. The higher probability

indicates a higher predicted likelihood of interaction. The code for sRNARFTarget nextflow pipeline is available at Github.

sRNARFTarget & IntaRNA			
	Escherichia coli	*Synechocystis*	*Pasteurella multocida*
No. of sequences in sRNA fasta file	22	2	1
No. of sequences in mRNA fasta file	4240	3179	1804

Table 3.6: Count of benchmarking sequences for sRNARFTarget & IntaRNA

3.5.2.2 IntaRNA

We downloaded IntaRNA source code from [72], installed it locally, and executed it from the command line. The data for running IntaRNA was the same as for sRNARFTarget as shown in Table 3.6. To obtain a total execution time for IntaRNA, we created a nextflow pipeline to run IntaRNA's two steps: getting the interaction energy and getting the p-values for the interaction energy. This pipeline has two processes. The first process takes two fasta files, one for sRNA and one for mRNA sequences and generates a CSV file containing the interaction energy for pairs along with other columns. The second process takes the file generated from the first one and runs an R script [73] and generates a file containing p-values and FDR values

calculated from energy. We used p-values as IntaRNA scores. IntaRNA results did not contain interaction energies/p-values for a few of the pairs. Final results had p-values for 92449 pairs in *E. coli*, 6344 pairs for *Synechocystis* and, 1803 pairs with p-values for *P. multocida*. The lower p-value indicates a higher predicted likelihood of interaction. Nextflow pipeline for running IntaRNA can be seen on Github.

3.5.2.3 CopraRNA

We ran CopraRNA from its webserver, [CopraRNA webserver link]. We took available pre-computed results for *E. coli* sRNAs and submitted jobs for the ones which were not available as pre-computed results on CopraRNA webserver. For *P. multocida* and *Synechocystis* we submitted jobs on the webserver with homologs as mentioned in section 3.5.1. We used the same parameter values for running CopraRNA as those used in the pre-computed results. We considered the result file containing p-values for all pairs. CopraRNA results did not contain interaction p-values for a few of the pairs. The result included p-values for 75841 pairs for *E. coli*, 5474 pairs for *Synechocystis*, and 1485 pairs for *P. multocida*. The lower p-value indicates a higher predicted likelihood of interaction.

3.5.3 Results standardization

We carried the following steps on the results of all three bacteria and the summary of final predictions of the programs can be found in Table 3.7.

1. sRNARFTarget predictions

    - Score (prediction probabilities) of sRNARFTarget are already rounded to five decimals.
    - We assigned corresponding classes to all predictions. As listed in Table 3.2 Benchmarking data, we assigned, class 1 (true positives) to 102 *E. coli* predictions, 22 *P. multocida* predictions and 20 *Synechocystis* predictions. Remaining all predictions were assigned with class 0 becoming true negatives.
    - We converted sRNA and mRNA IDs to lowercase.

2. IntaRNA predictions

    - As mentioned earlier, lower p-values indicate more likely predicted interaction. We subtracted IntaRNA p-values from 1, to make it consistent with our program so that p-values now become predicted interaction probability. We then rounded the score to five decimals.
    - Assigned target classes 1 and 0 as mentioned in sRNARFTarget steps.
    - Converted sRNA and mRNA IDs to lowercase.

3. Intersection between sRNARFTarget & IntaRNA predictions

    - IntaRNA did not generate predictions for all the given input pairs. Hence, we wrote an R script to get the common pairs predicted by both programs

so that the results are consistent across the programs. Common pairs are henceforth referred to as RI pairs. As a result of this, we got 92449 for *E. coli*, 6344 for *Synechocystis* and, 1803 for *P. multocida*, pairs with scores for sRNARFTarget and IntaRNA.

4. CopraRNA predictions

   - We used the CopraRNA result file that has predictions for all pairs. We removed rows where p-values were empty or NA.
   - We noticed that there were duplicates in CopraRNA predictions: there were a number of sRNA-mRNA pairs that had two entries with two different p-values. So to eliminate the duplicate entries, we wrote an R script to get the most significant p-value (lowest p-value) for each sRNA-mRNA pair, and remove all other entries.
   - CopraRNA lower p-values indicate more likely predicted interaction. We subtracted p-values from 1 as we did for IntaRNA, to obtain predicted interaction probability. We rounded predicted interaction probabilities to five decimals.
   - Assigned target class 1 and 0 to predictions accordingly as for the other two programs.
   - Converted sRNA and mRNA IDs to lowercase.

5. Intersection between CopraRNA predictions and RI pairs.

- Like IntaRNA, CopraRNA also did not generate predictions for all the pairs. We first extracted common pairs between CopraRNA pairs and RI pairs (intersection between sRNARFTarget and IntaRNA). Next, we extracted the pairs in RI pairs that are not present in the CopraRNA result and assigned those with a predicted interaction probability equal to 0. This makes the count of predictions the same across all three programs. As a result of this, we got 92449 for *E. coli*, 6344 for *Synechocystis* and, 1803 for *P. multocida*, predictions with scores for sRNARFTarget, CopraRNA and IntaRNA.

6. Sorted each program's predictions by their score in descending order.

Count/Bacteria	*Escherichia coli*	*Synechocystis*	*Pasteurella multocida*
No. of sRNAs	22	2	1
No. of sRNA-mRNA pairs	92449	6344	1803
True positives	101	20	22
True negatives	92348	6324	1781

Table 3.7: Final benchmarking dataset used for all three programs

3.5.4 Plots

We created precision-recall curves (PR) and receiver-operating characteristic curves (ROC), Violin box plots and line plots for comparing the performance of sRNATarget, CopraRNA and IntaRNA for all three bacteria. Here we will discuss the functions/modules or approach we used to create these plots and the plots can be seen in Chapter 4.

3.5.4.1 PR & ROC plots

We used the PRROC [74] package from R to plot PR and ROC curves. We plotted individual plots for each bacterium per program and combined plots containing three curves showing three programs per bacterium.

3.5.4.2 Violin Box plots

We created three violin box plots [75], each per bacterium containing three box plots for each program. For box plots, we first ranked the predictions of each program. We used the Ordinal ranking (ties are given consecutive ranks) system and used python's Ranking [76] module to get the ranks for predictions. Rank 1 corresponds to the prediction with the highest predicted interaction probability. We then took the ranks corresponding to the true positives and plotted the violin box plots using the ggplot2 [77] R package.

We ran a Mann-Whitney test [78] using wilcox.test command from [79] R for true

positives with ordinal ranks to a pair-wise statistical comparison of the programs ranks. We included the Mann-Whitney test p-values in the corresponding violin box plots.

3.5.4.3 Line plots

To create the line plots, we took the top 10% predictions for each program, counted the number of true positives, and calculated the percentage of true positives among the top 10% of predictions. Then iteratively increase the percentage of top predictions by 10% and repeat the process described above until all predictions (100%) are taken. We plot the percentage of predictions on the x-axis and percentage of true positives on the y-axis.

3.6 sRNARFTarget interpretability program

We created two python scripts for sRNARFTarget interpretability using SHAP and pyCeterisParibus python packages. See Figure 3.4 for the workflow of the programs. Both scripts can be run from the command line. Instructions on running these programs can be seen on Github.

To run SHAP for a sRNARFTarget prediction, the user can choose an sRNA-mRNA pair of interest from Prediction_probabilities.csv file under sRNARFTargetResult folder, generated by sRNARFTarget program. This pipeline takes two command-line arguments, sRNA ID and mRNA ID. sRNA and mRNA IDs have to be the

same as in the Prediction_probabilities.csv file. It then fetches the features for chosen sRNA-mRNA pair from 'FeatureFile.csv' file under sRNARFTargetResult folder. This program uses TreeExplainer of SHAP to create the explainer. We used TreeExplainer as the underlying model is a Random forest. Then it calculates the SHAP values for a given observation (sRNA-mRNA pair). Lastly, it generates SHAP's decision, waterfall and force plots for interpretation.

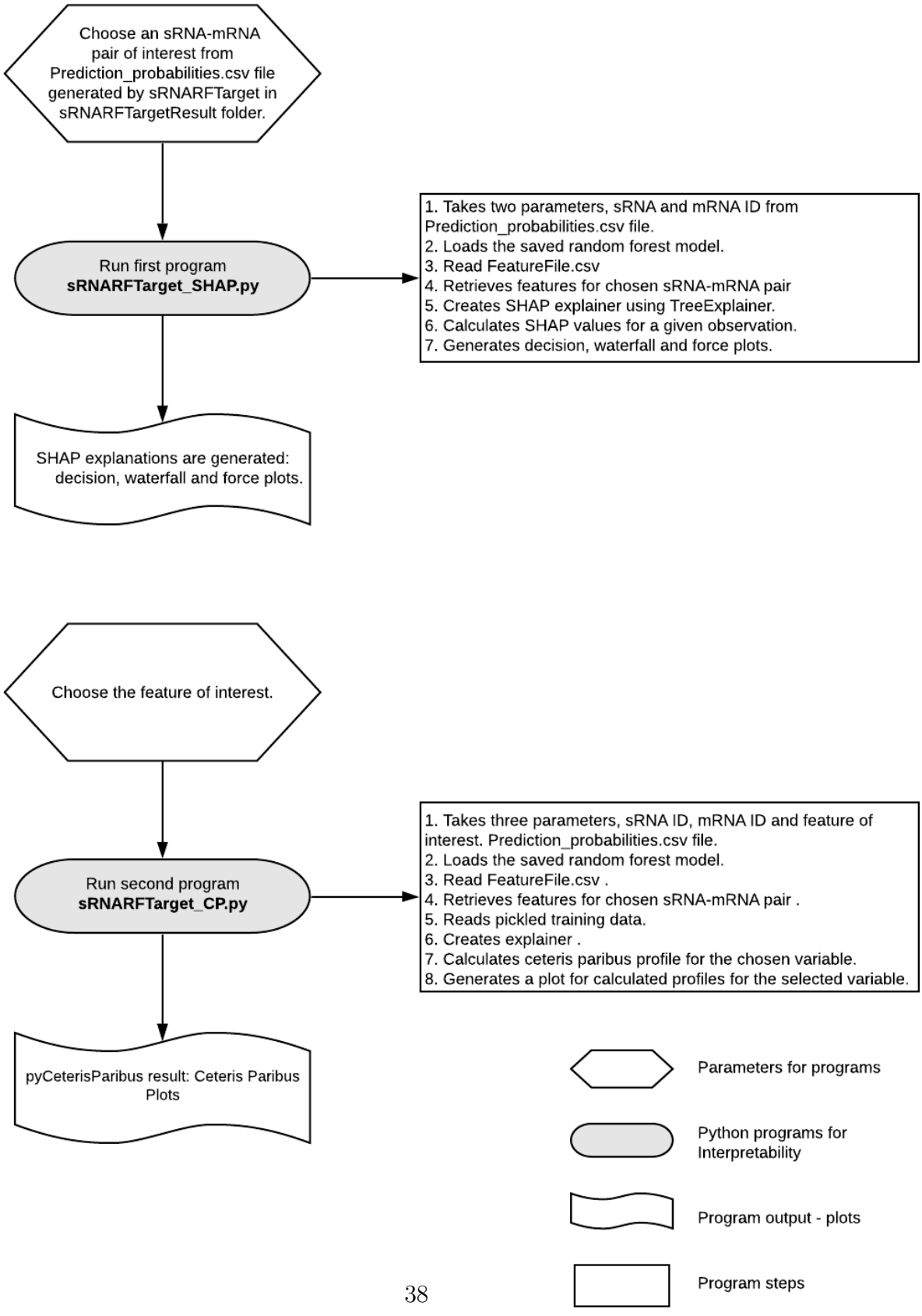


Figure 3.4: sRNARFTarget interpretability workflow

By looking at the SHAP plots, the user can select a variable of interest to run the pyCeterisParibus program. This program takes three arguments, sRNA ID, mRNA ID (same which were passed in SHAP) and feature name. Then it creates the explainer using training data and calculates ceteris paribus profiles for a chosen variable for given sRNA-mRNA pair. It fetches the features for chosen sRNA-mRNA pair from 'FeatureFile.csv' file under sRNARFTargetResult folder and plots the calculated profiles for the chosen variable.

3.7 Summary

In this chapter, we discussed the data collection. Using the nextflow pipelines, we extracted the sequences and features; nucleotide frequency and secondary structure distance. We created true negatives by permuting the true positives. We did Grid search CV for selecting the best estimator for models generated by random forest, KNN, and gradient boosting, and used stratified cross-validation for training the models using the training data. We first trained these models using the small training data 102 *E. coli* pairs with four sets of features (Trinucleotide frequency, Trinucleotide frequency difference, Tetranucleotide frequency, Tetranucleotide frequency difference of sRNA-mRNA sequences). Based on the results, we created models with trinucleotide frequency difference and tetranucleotide frequency difference with entire training data 745 pairs. We retrieved sRNA-mRNA secondary structure distances to see whether these features increase the model's performance. Adding distance features increased

the execution time from seconds to hours and did not increase the performance of any model. Based on our analysis and results, we selected a random forest model trained with trinucleotide frequency difference. We then created the sRNARFTarget nextflow pipeline that uses the random forest model for prediction. We considered CopraRNA and IntaRNA for benchmarking. We ran CopraRNA from webserver, IntaRNA from the command line and sRNARFTarget from nextflow pipeline, with benchmarking data containing sRNA-mRNA sequences for three bacteria; *E. coli*, *P. multocida* and *Synechocystis*.

We implemented two python programs to facilitate understanding sRNARFTarget predictions using SHAP and pyCeterisParibus python packages. From the results obtained from sRNARFTarget for given sequences, one can choose the sRNA-mRNA pair of interest and get the visual interpretations from sRNARFTarget_SHAP.py and by looking at these plots, one can also optionally choose the feature and get the interactive plot by running sRNARFTarget_CP.py, which shows the real-time change in the prediction as the value of the chosen feature changes.

Chapter 4

Results and Discussion

In this chapter, we will discuss and present the results obtained first in a small pilot experiment and then in experiments using the complete training data. At the end of the chapter, we present the results of comparatively assessing the performance of three programs for sRNA target prediction (sRNARFTarget, CopraRNA, and IntaRNA) on data not seen by sRNARFTarget during training.

4.1 Pilot experiment using small training data

The pilot experiment allows us to choose sequence-derived features to infer sRNA targets with a performance comparable to other non-comparative genomics sRNA target prediction software. Table 4.1 shows the AUROC and average precision scores for small data (102 *E. coli* pairs) for trinucleotide and tetranucleotide frequency of sRNA-mRNA pairs. We generated 102 true negative instances to get a balanced

training data set. The pilot study only includes data from a single bacterium and roughly 14% of the sRNA-mRNA pairs in the full data set. The performance was less than random performance (0.5 AUROC) for all three models.

	AUROC (mean)				**Average precision (mean)**			
	All features		Features selected: MDA >0		All features		Features selected: MDA >0	
Classifiers	Tri nt.	Tetra nt.	Tri nt.	Tetra nt.	Tri nt.	Tetra nt.	Tri nt.	Tetra nt.
RF	0.17	0.05	0.15	0.13	0.34	0.34	0.36	0.36
KNN	0.48	0.5	0.39	0.39	0.54	0.56	0.46	0.47
GB	0.12	0.12	0.19	0.21	0.36	0.35	0.38	0.38

Table 4.1: AUROC and average precision score obtained in a pilot experiment on a small training data (102 *E. coli* pairs) using trinucleotide and tetranucleotide frequency as features.

Table 4.2 shows the AUROC and average precision scores for small data for trinucleotide frequency difference and tetranucleotide frequency difference of 102 *E. coli* sRNA-mRNA pairs. Using the difference between nucleotides frequency as features caused an increase in the model performances. KNN also has better performance with trinucleotide frequency difference.

We adopted the idea of using sequence-derived features such as nucleotides fre-

quency from previous studies such as [80] and [81] that use k-mer frequency for predicting long non-coding RNAs, [82] that uses k-mer composition for DNA sequence classification and [83] that makes use of k-mer composition for predicting small non-coding RNAs. As sRNAs bind mRNAs through base pairing [84], then taking nucleotide frequency difference might capture this for the classifiers to use. Therefore we then included trinucleotide and tetranucleotide frequency difference as features. We started with trinucleotide composition, and as the performance was not substantially increasing with tetranucleotide composition, thereby, we decided not to go beyond tetranucleotide composition.

	AUROC (mean)				**Average precision (mean)**			
	All features		Features selected: MDA >0		All features		Features selected: MDA >0	
Classifiers	Tri nt. Diff	Tetra nt. Diff	Tri nt. Diff	Tetra nt. Diff	Tri nt. Diff	Tetra nt. Diff	Tri nt. Diff	Tetra nt. Diff
RF	0.5	0.11	0.5	0.12	0.58	0.35	0.59	0.35
KNN	0.53	0.44	0.54	0.45	0.54	0.49	0.57	0.52
GB	0.49	0.27	0.52	0.24	0.57	0.43	0.57	0.39

Table 4.2: AUROC and average precision score for a small training data (102 *E. coli* pairs) for trinucleotide and tetranucleotide frequency difference.

When using all features, the performance of random forest in terms of AUROC increased from .17 with trinucleotide composition to .5 with trinucleotide frequency difference, gradient boosting got increased from .12 to .49 and KNN from .48 to .53. Tetranucleotide frequency difference did not increase the performance as trinucleotide difference did, and reduced AUROC in KNN by 6%. Random forest and KNN surpassed random performance with trinucleotide frequency difference. As removing features based on their MDA does not substantially affect the models' performance, we decided to use all features available.

As per the results from [1] for the same dataset, AUROC for CopraRNA was 0.46 and for IntaRNA, 0.27. As our performance was comparable to CopraRNA performance, we decided to continue with sequence-derived features such as trinucleotide frequency difference that allows us to distinguish interacting sRNA-mRNA pairs. Thus, we proceeded to train with a larger dataset.

As all models' performance was lower than random performance when trinucleotide and tetranucleotide composition were used as features, we discarded these from further experiments.

4.2 Using a larger training data set improves model performance

The complete training data consists of 745 interacting sRNA-mRNA pairs (true positives), and 745 random sRNA-mRNA pairs (true negatives). Table 4.3 shows the performances of the best models per classifier when trained on the entire training data using trinucleotide frequency difference and tetranucleotide frequency difference as features. Performances achieved with trinucleotide frequency difference was better than tetranucleotide frequency difference and substantially higher for random forest and gradient boosting. With trinucleotide frequency difference, the model with the best performance was the random forest, followed by gradient boosting and then KNN. Using the larger training data (745 pairs) instead of the small training data (102 pairs) improved the performance of random forest from .5 to .67 AUROC.

RNA secondary structures are associated with the regulation of mRNA [85]. [86] used RNA secondary structure level information for a visualization method of the prediction of sRNA-mRNA interaction. [83] used similar (structural motifs) to predict non-coding RNAs. Also, since the secondary structure of both sRNA and mRNA affects their binding [87], we decided to include secondary structure distances as features together with the tri-(tetra-)nucleotide frequency difference. Models performance with trinucleotide frequency difference with secondary structure distances and tetranucleotide frequency difference with secondary structure distances are in Table

	AUROC (*mean* ± *std*)		**Average precision** (*mean* ± *std*)	
Classifiers	Tri nt. Diff	Tetra nt. Diff	Tri nt. Diff	Tetra nt. Diff
RF	0.67 ± 0.03	0.05 ± 0.02	0.65 ± 0.04	0.31 ± 0
KNN	0.63 ± 0.03	0.01 ± 0	0.60 ± 0.02	0.45 ± 0.01
GB	0.66 ± 0.03	0.06 ± 0.02	0.62 ± 0.03	0.32 ± 0

Table 4.3: 10-fold CV AUROC and average precision score for the best model per classifier trained on sequence-derived features (trinucleotide composition and tetranucleotide composition) from 745 sRNA-mRNA pairs.

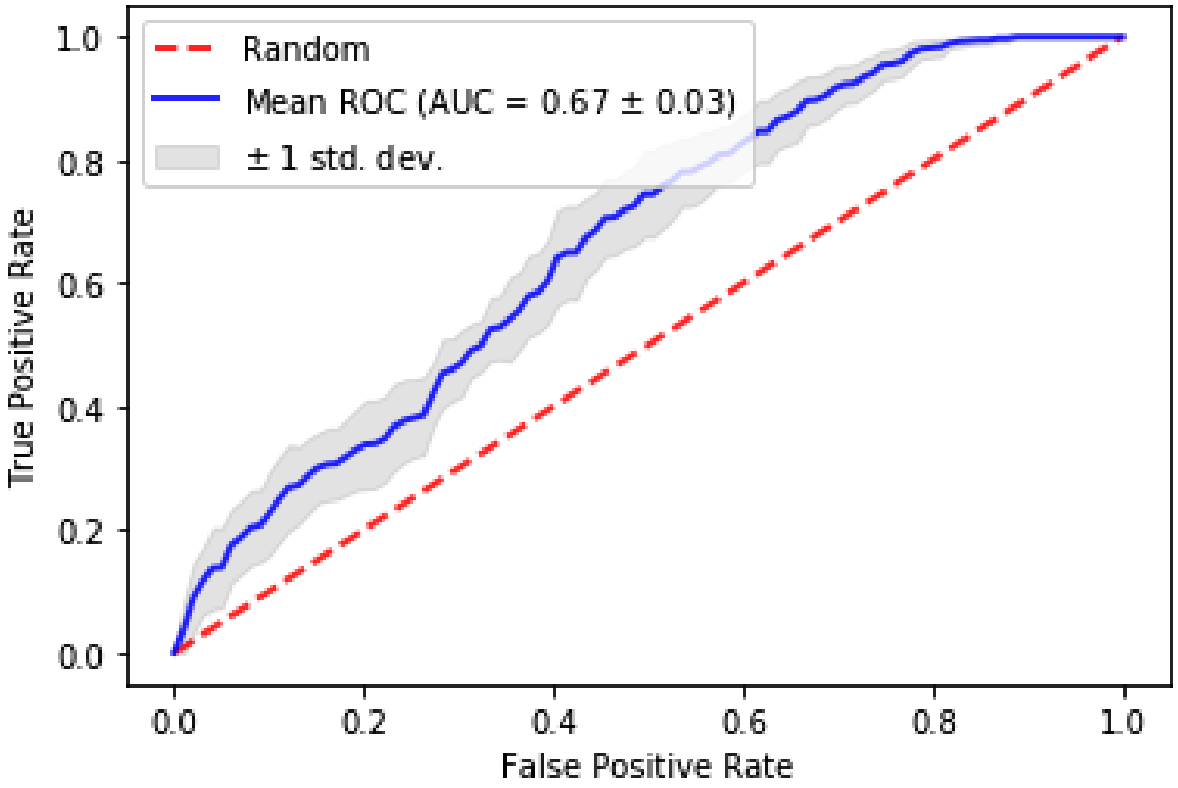
4.4. With trinucleotide frequency difference with secondary structure distances, the performance was unchanged for random forest. It was dropped by more than half for KNN and went slightly up for gradient boosting. Adding secondary structure distance features with tetranucleotide frequency difference features had little to no effect.

	AUROC ($mean \pm std$)		**Average precision** ($mean \pm std$)	
Models	Tri nt. Diff & Dist.	Tetra nt. Diff & Dist.	Tri nt. Diff & Dist.	Tetra nt. Diff & Dist.
RF	0.67 ± 0.03	0.04 ± 0.01	0.65 ± 0.04	0.31 ± 0
KNN	0.27 ± 0.04	0.25 ± 0.06	0.38 ± 0.02	0.38 ± 0.03
GB	0.67 ± 0.04	0.08 ± 0.02	0.64 ± 0.04	0.32 ± 0

Table 4.4: 10-fold CV AUROC and average precision score for the best model per classifier trained on sequence-derived features (trinucleotide and tetranucleotide frequency difference with secondary structure distance) from 745 sRNA-mRNA pairs.

Adding distance features did not substantially improve models performance, and dramatically increased the execution time (from seconds to hours) to extract the features; hence, we dropped the distance features. (Note: Centroid fold program for fetching secondary structures of RNAs increased the execution time, RNAdistance program for getting the distance did not take a long time). Random forest and gradient boosting models were comparable in terms of AUROC and average precision; however, the random forest was much faster to train than gradient boosting. Thus, we decided to create our final model using random forest and included this model in the sRNARFTarget pipeline. Figure 4.1 shows the ROC curve and the average precision score for the final random forest model. The parameters to create this model are 500 trees (n_estimators), log2 of features for split (max_features), and the maximum

depth of the tree (max_depth) is 9.




```
mean Average Precision=0.65
std Average Precision=0.04
```

Figure 4.1: 10-fold CV ROC curve and average precision score for random forest model

Figure 4.2 shows the top 30 most important features for random forest model using trinucleotide frequency difference as features.

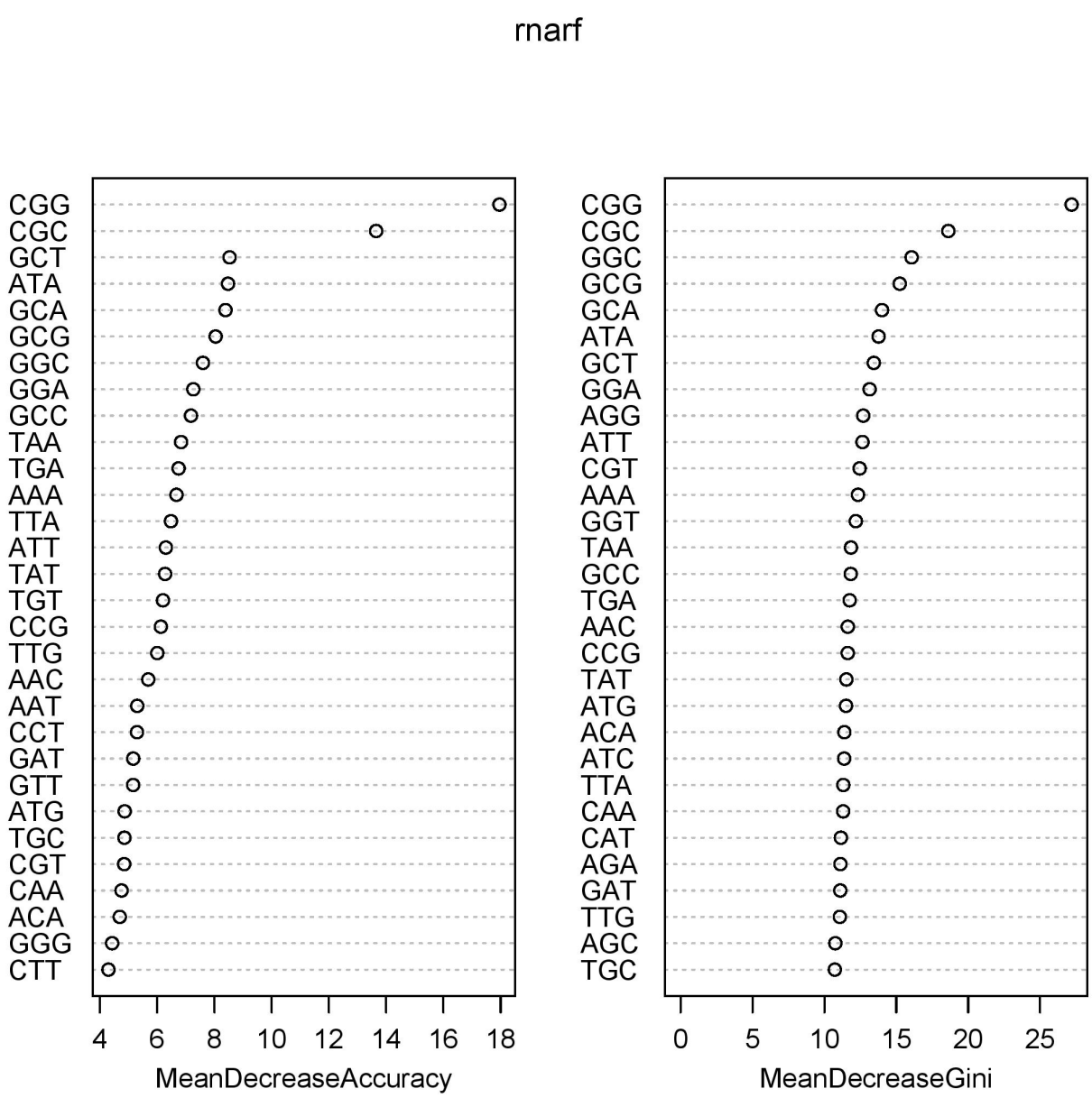


Figure 4.2: Feature importance plot obtained by R function for entire training data containing trinucleotide frequency difference.

4.3 Benchmarking results on independent data sets

We compared sRNARFTarget's performance with that of CopraRNA and IntaRNA on data from three bacteria *E. coli*, *Synechocystis* and, *P. multocida*. Table 4.5 shows the area under the ROC curve (AUROC) and area under the Precision-Recall curve (AUPRC) for all three programs per bacterium.

AUROC			
Bacteria/Program	**CopraRNA**	**sRNARFTarget**	**IntaRNA**
Escherichia coli	0.88	0.65	0.62
Synechocystis	0.95	0.63	0.48
Pasteurella multocida	0.65	0.44	0.40
AUPRC			
Escherichia coli	0.0767	0.0038	0.0018
Synechocystis	0.2335	0.0048	0.0037
Pasteurella multocida	0.0359	0.0105	0.0091

Table 4.5: The area under the ROC curve and PR curve for benchmarking data

Figure 4.3 and Figure 4.4 show the ROC and PR curves for *E. coli*. CopraRNA performed best among the three programs followed by sRNARFTarget then IntaRNA. Figure 4.5 and Figure 4.6 show the ROC and PR curve respectively for *Synechocystis*. Figure 4.7 and Figure 4.8 show the ROC and PR curves for *P. multocida*.

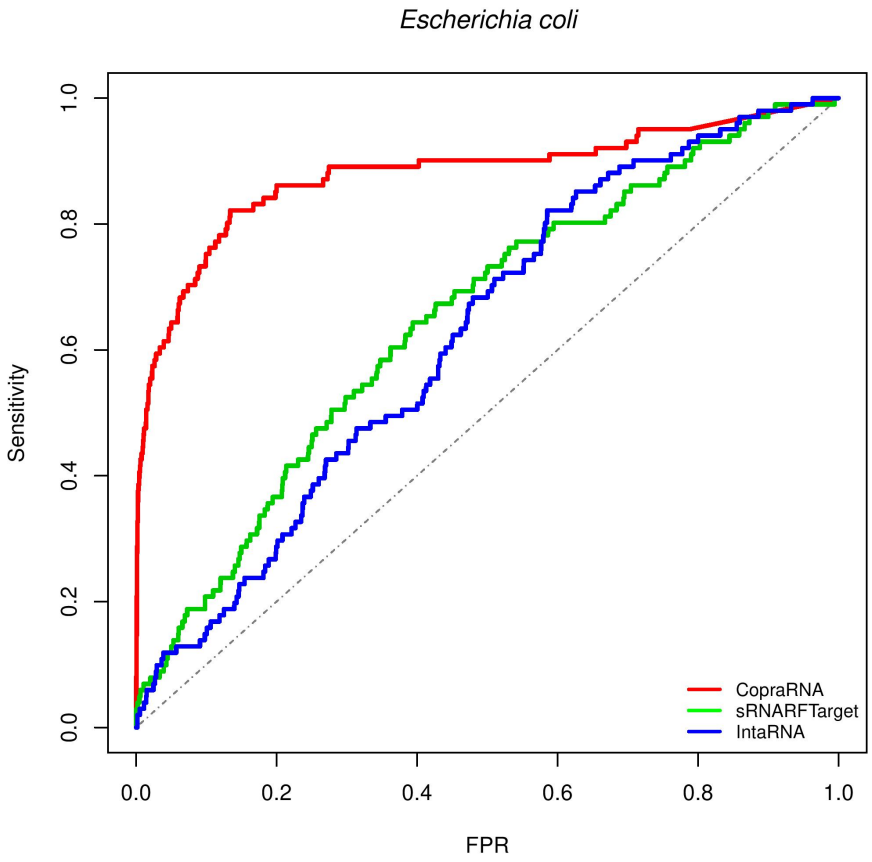


Figure 4.3: *Escherichia coli* ROC curve

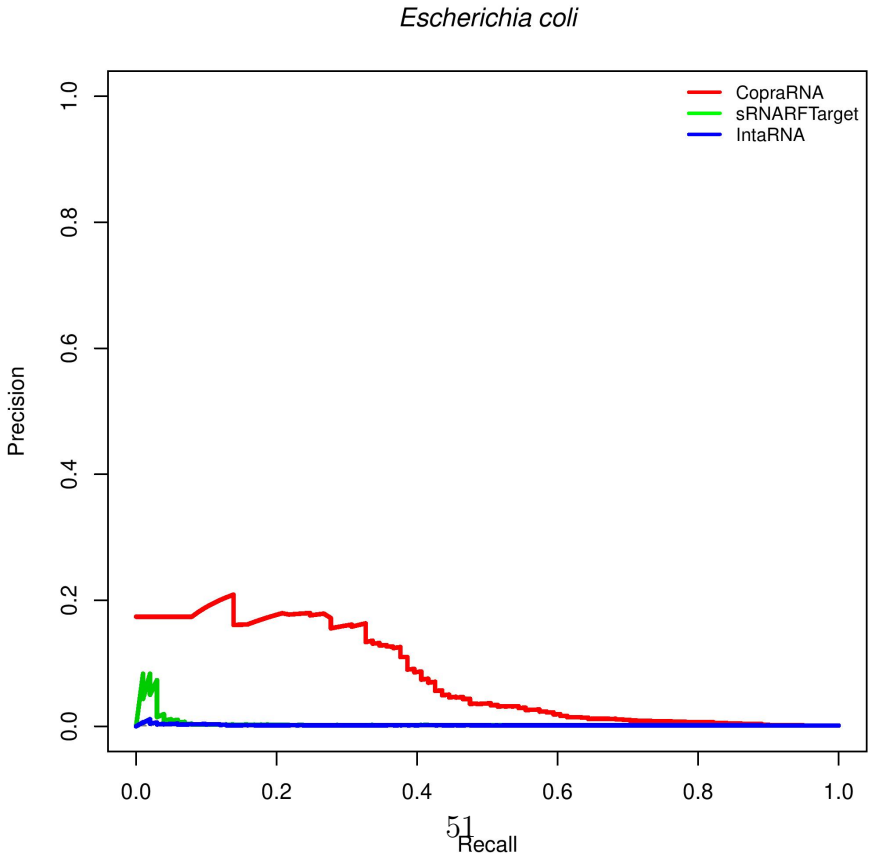


Figure 4.4: *Escherichia coli* PR curve

CopraRNA performs best for all three bacteria. CopraRNA is a comparative genomic-based approach and requires sequence conservation of the sRNAs and mRNAs in at least four bacteria. sRNARFTarget outperforms IntaRNA, which was the best non-comparative genomic-based approach as shown by [1].

sRNARFTarget does not require sequences to be conserved in other bacteria and overcomes this limitation sequence conservation. sRNARFTarget can be run for any number of sRNAs and mRNAs at a time. sRNARFTarget overcomes the limitation of running the program for one sRNA at a time of CopraRNA. sRNARFTarget uses the whole sequences of sRNA and mRNA, and CopraRNA uses untranslated regions (UTR).

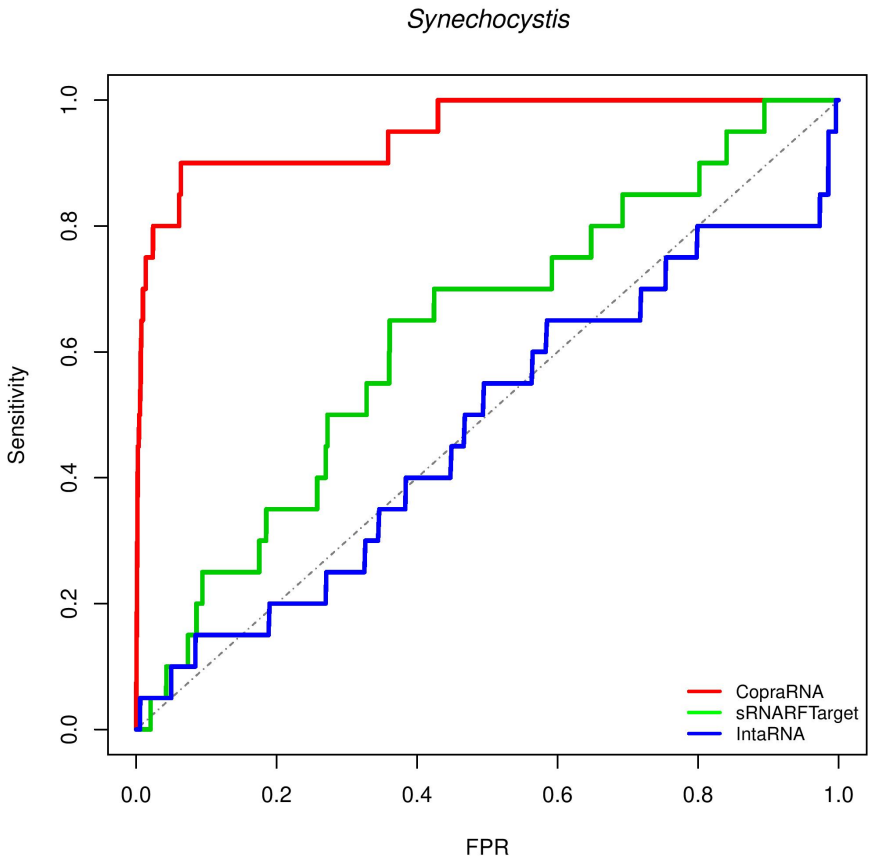


Figure 4.5: *Synechocystis* ROC curve

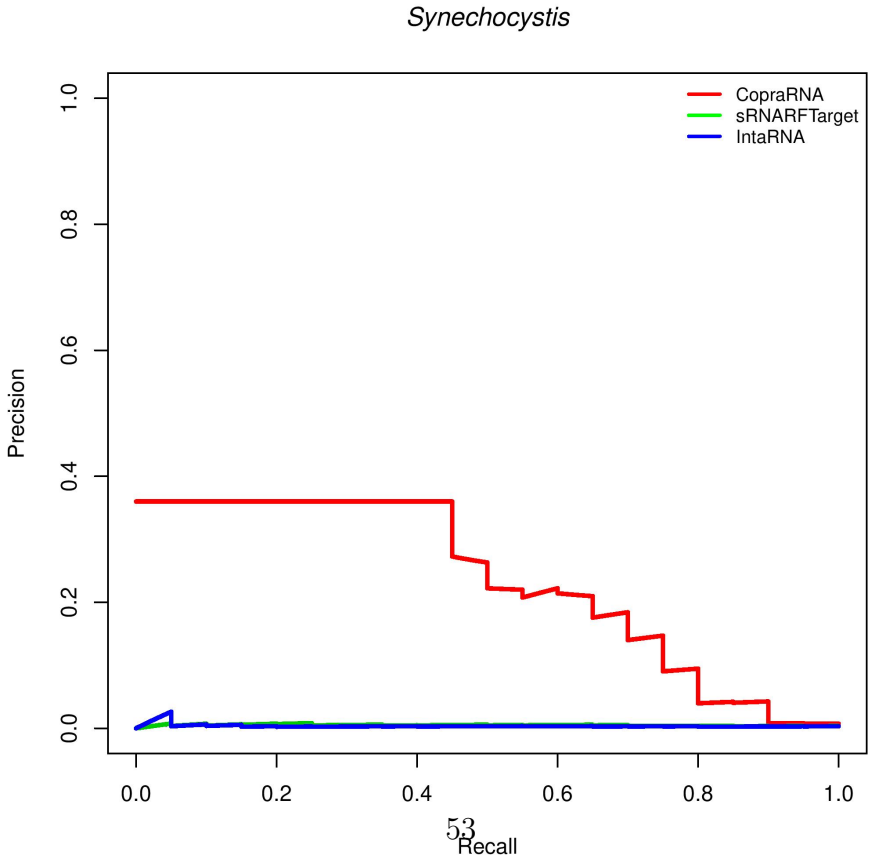


Figure 4.6: *Synechocystis* PR curve

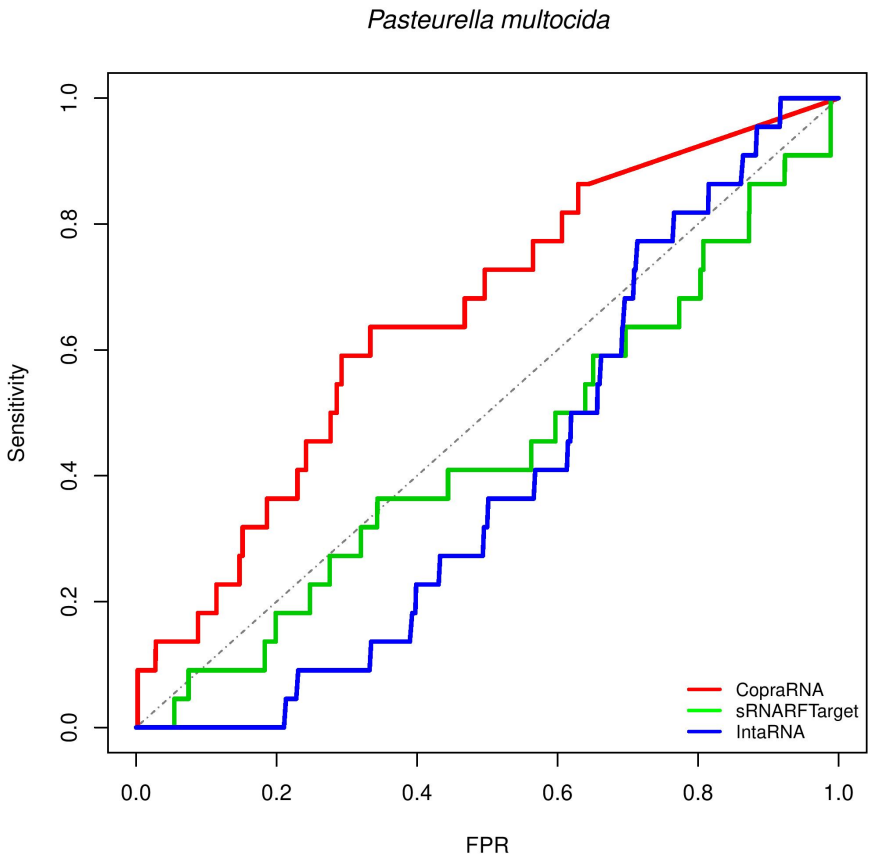


Figure 4.7: *Pasteurella multocida* ROC curve

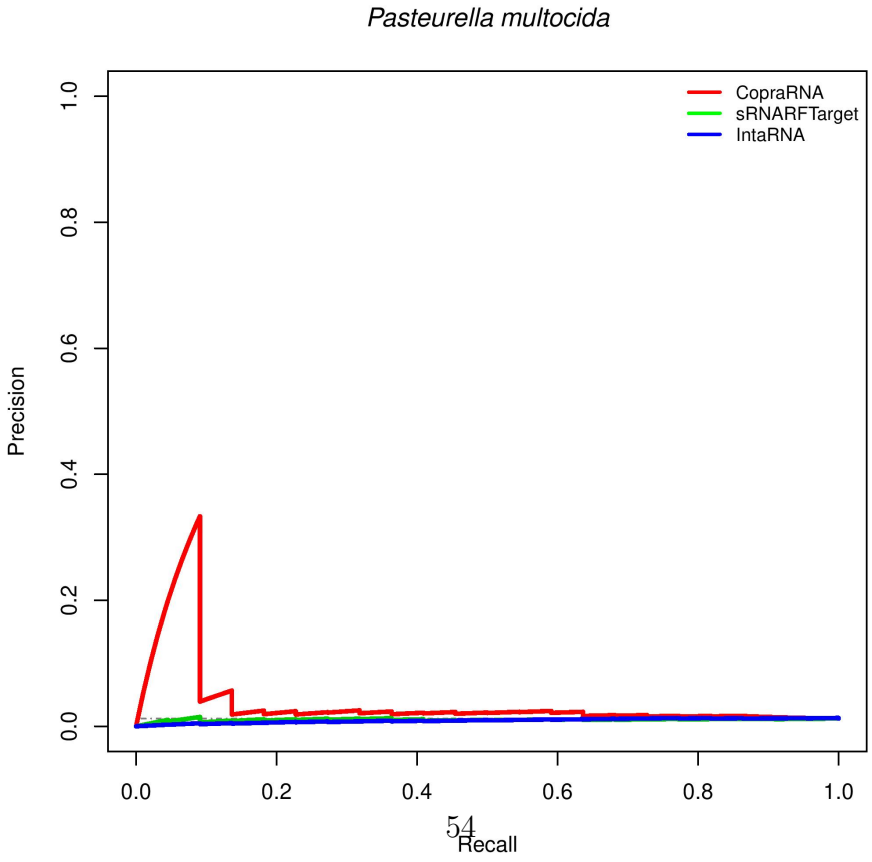


Figure 4.8: *Pasteurella multocidas* PR curve

Figure 4.9 shows the violin box plot for *E. coli*. The violin box plot shows the rank distribution of true positives and the shape surrounding the box plots shows the data density for different values. The horizontal bar in the box shows the median rank of the true positive. CopraRNA has a lower median rank followed by sRNARFTarget and then IntaRNA. A lower rank indicates that the program predicts with higher confidence the true positive pairs as interacting pairs. The shape of CopraRNA suggests that most of the true positives are ranked before most non-interacting pairs. The shape of the plot for sRNARFTarget is a bit wider at the bottom than the top. It has more true positives at the bottom (with lower rank) than the top and narrows at the top. IntaRNA has the true positives over three-fourths of its rank range and narrows down as it goes up.

The p-values obtained from the Mann-Whitney test are shown in Figure 4.9. These p-values indicate that CopraRNA's median rank of true positives is significantly lower than sRNARFTarget's median rank of true postives. Note that the lowest or top rank is 1. Similarly, sRNARFTarget assigns significantly lower ranks to true positives than IntaRNA.

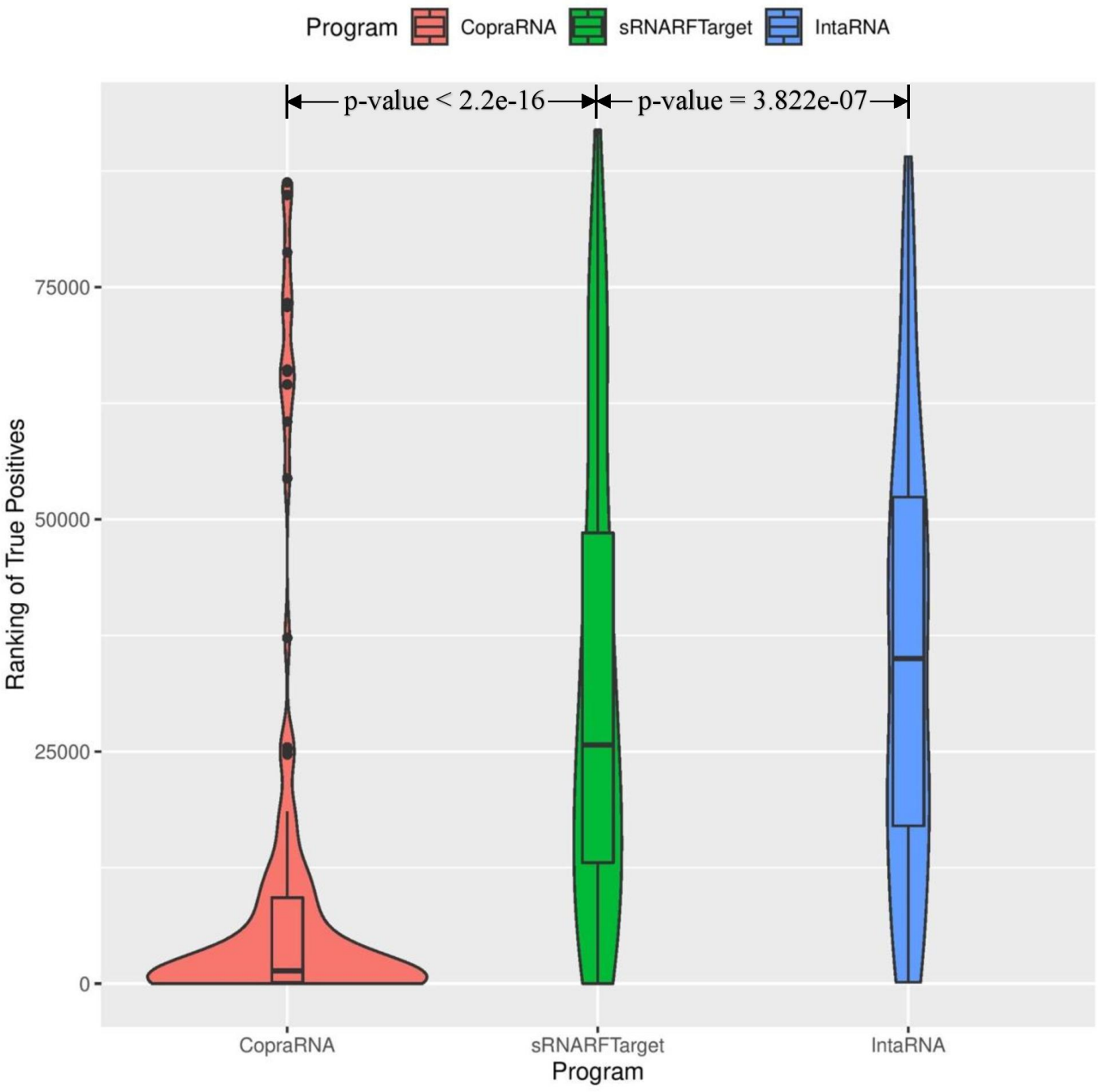


Figure 4.9: Violin box plot for *Escherichia coli*

Figure 4.10 and Figure 4.11 show the violin box plots for *Synechocystis* and *P. multocida* respectively. For these two bacteria as well, the median rank is lower in CopraRNA followed by sRNARFTarget and IntaRNA.

Figure 4.10 also shows the p-values from the Mann-Whitney test. P-value for CopraRNA-sRNARFTarget program pair is 4.778e-05. Hence, it suggests that the median rank of the true positive ranks of the two programs is notably different. The median rank of true positives assigned by sRNARFTarget is significantly lower than the ranks of true positives assigned by IntaRNA with a p-value of 4.768e-06.

All three programs found more difficult to distinguish true interacting pairs in *P. multocida* ranking true positives with higher ranks (Figure 4.11). Nevertheless CopraRNA still ranks true positives significantly lower than sRNARFTarget (p-value = 2.15e-05), and sRNARFTarget ranks true positives lower than IntaRNA (p-value = 0.056).

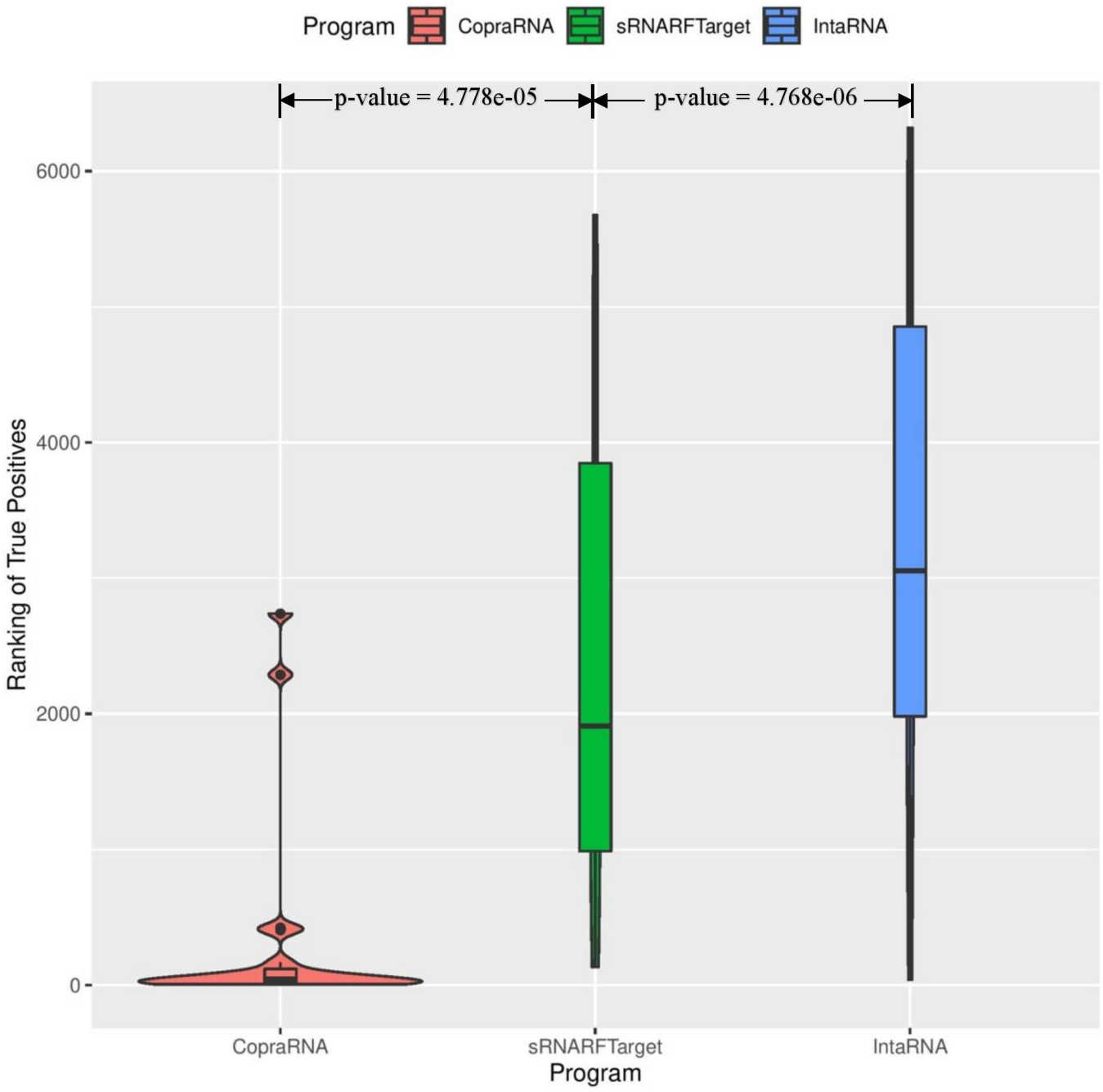


Figure 4.10: Violin box plot for *Synechocystis*

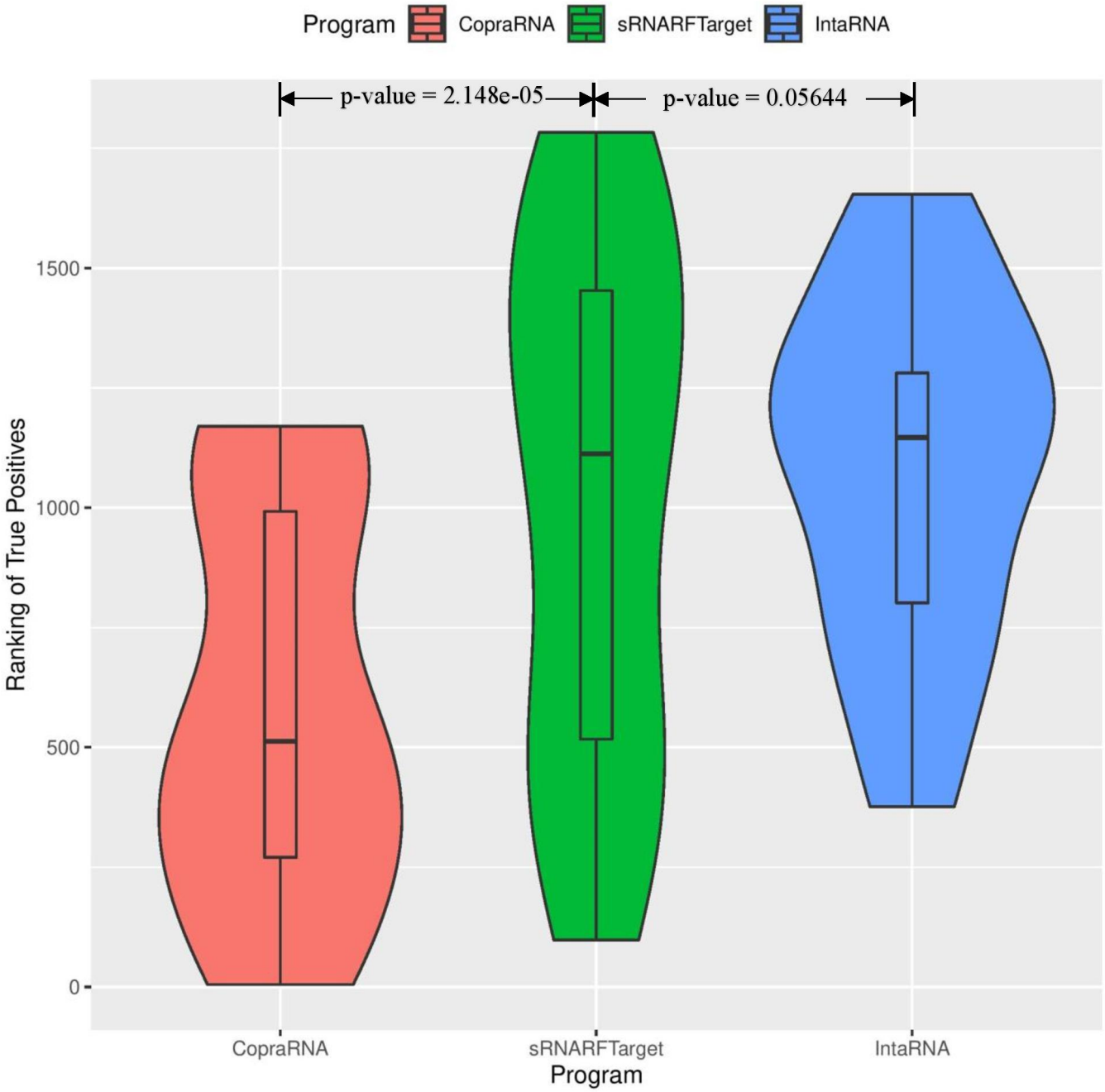


Figure 4.11: Violin box plot for *Pasteurella multocida*

Lastly, we plotted the line plots to get the percentage of true positives predicted among a certain percentage of predicted interacting pairs. Figure 4.12 shows the percentage plot for *E. coli*. In the top 10% predictions, CopraRNA predicted 74% true positives, sRNARFTarget predicted 21% true positives, and IntaRNA predicted 14% true positives. In terms of the percentage of true positives in all the predictions, this plot suggests that after CopraRNA, sRNARFTarget outperformed IntaRNA.

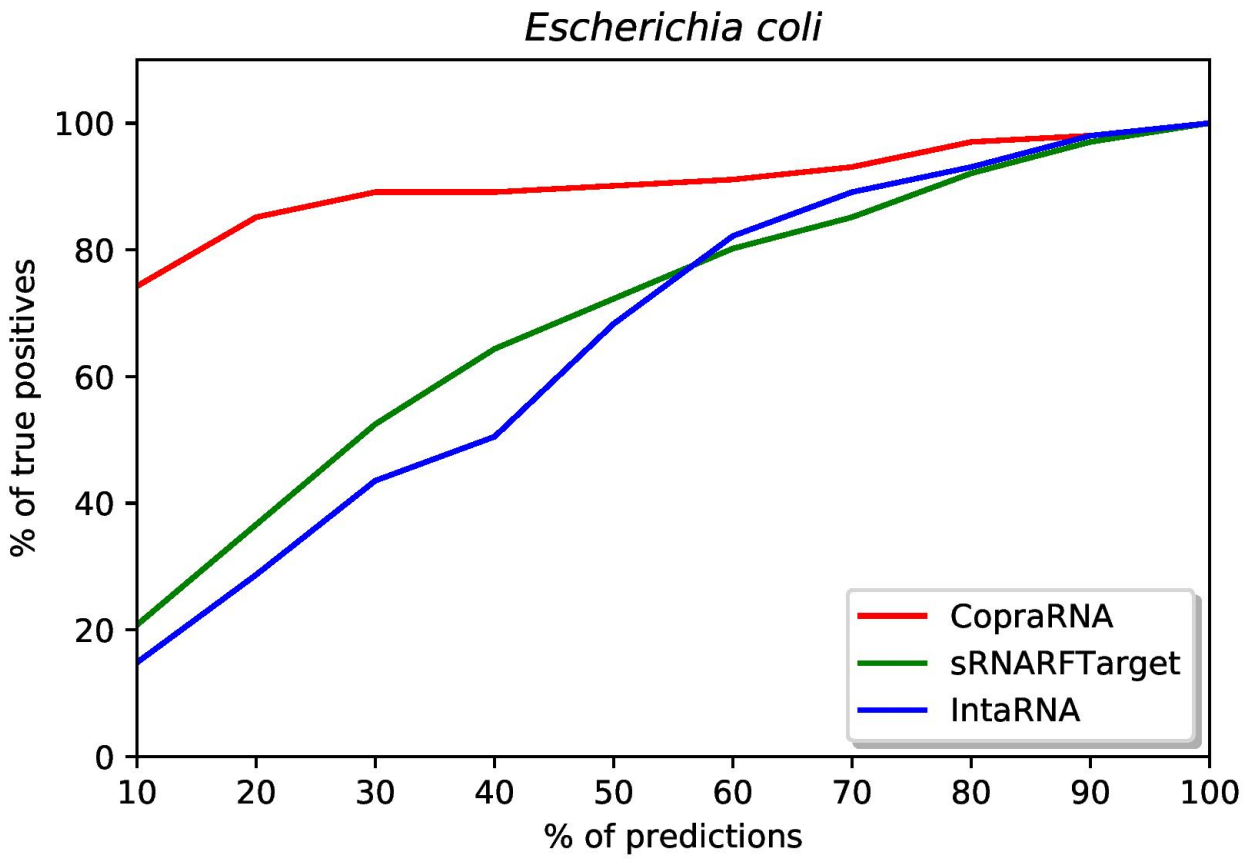


Figure 4.12: Percentage plot for *Escherichia coli*

Figure 4.13 shows the percentage plot for *Synechocystis*. From the top 50% predictions, CopraRNA predicted 100% true positives, sRNARFTarget predicted 70% true positives and IntaRNA predicted 55% of true positives.

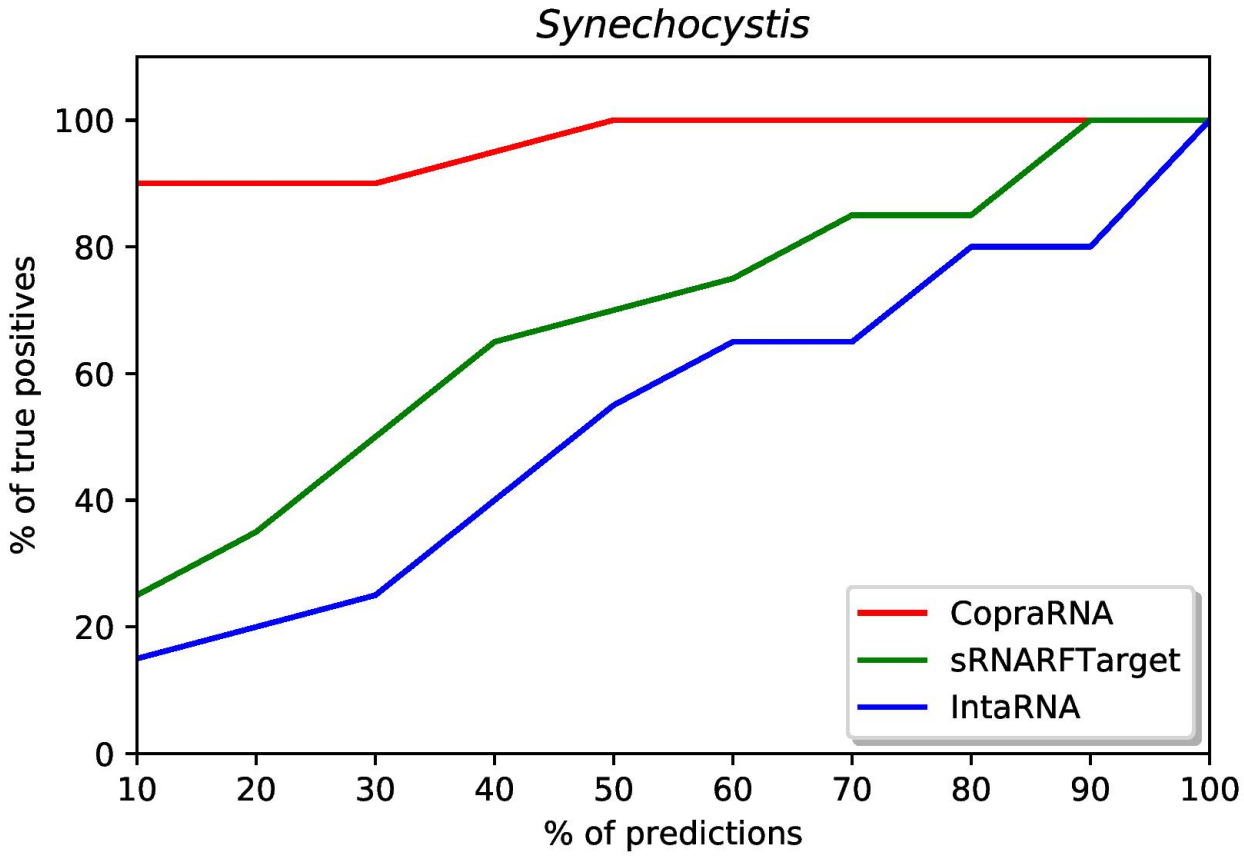


Figure 4.13: Percentage plot for *Synechocystis*

Figure 4.14 shows the percentage plot for *P. multocida*. In top 20% predictions, CopraRNA predicted 18% of true positives. sRNARFTarget was able to predict 10% of true positives. IntaRNA did not predict any true positives in top 20% predictions.

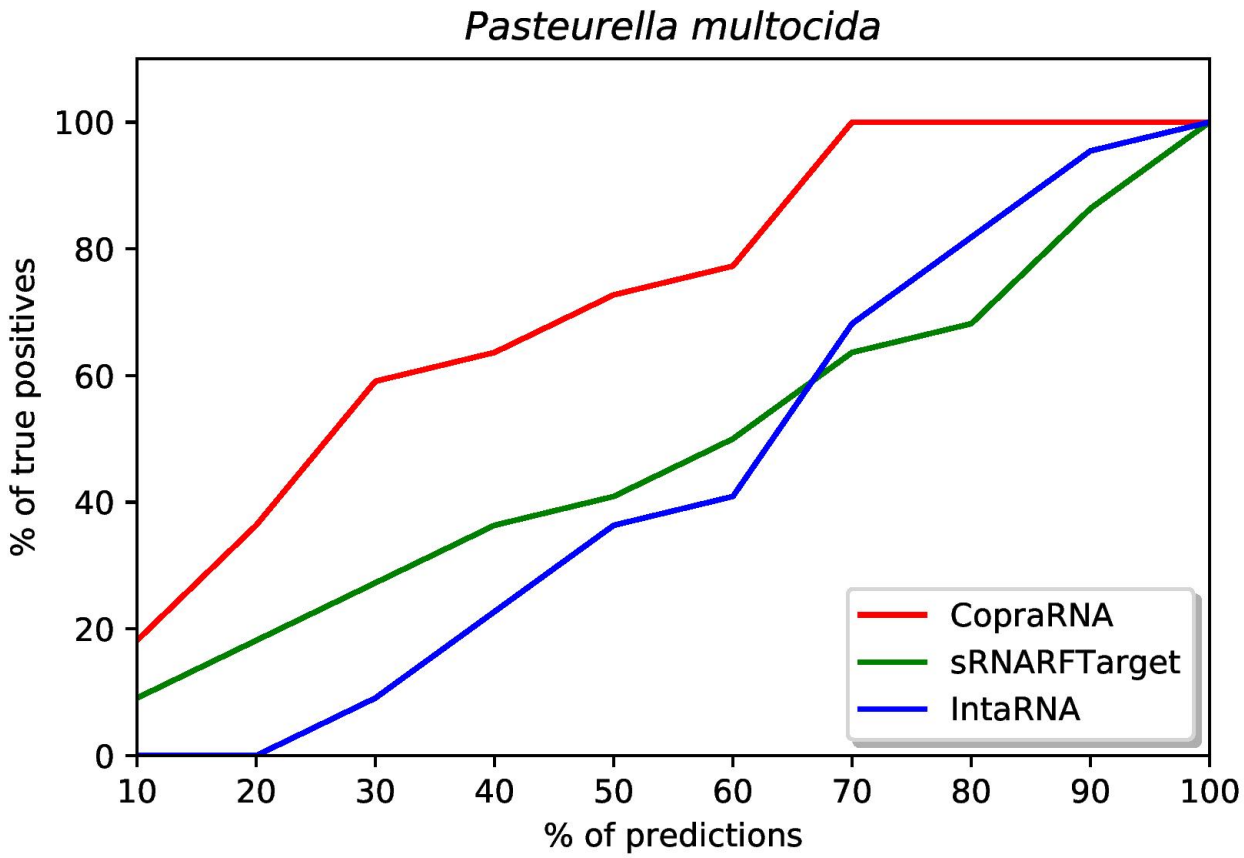


Figure 4.14: Percentage plot for *Pasteurella multocida*

In terms of execution time, sRNARFTarget is faster than CopraRNA and IntaRNA. For 1804 sRNA-mRNA pairs of *P. multocida*, sRNARFTarget took 31.4 seconds and IntaRNA took 1 hour, 43 minutes and 16 seconds. Table 4.6 shows the execution time for all three bacteria for IntaRNA and sRNARFTarget. Table 4.7 shows the time taken by the CopraRNA webserver for job completion. These times were calculated by taking the difference between the job submission time and the job completion time (timestamp of job completion email). These times are not compara-

ble to that of Table 4.6 as CopraRNA was run from webserver and sRNARFTarget and IntaRNA were run from the command line. The decrease in running time observed in sRNARFTarget might be due to the fact that CopraRNA starts with a genome-wide target prediction for each organism considered. Then it combines the predictions for homologous targets in all organisms. IntaRNA calculates the free energy needed to make the interaction site accessible. It also calculates Hybridization energy to find the quality of an RNA–RNA interaction between the target sites. While sRNARFTarget calculates the trinucleotide frequency difference and uses a random forest, which is a collection of decision trees. Decision trees are fast to provide a prediction.

Bacteria	**No. of pairs**	**Execution time**	
		sRNARFTarget	**IntaRNA**
Pasteurella multocida	1804	31.4s	1h 43m 16s
Synechocystis	6358	1m 18s	2h 33m 2s
Escherichia coli	93280	15m 56s	3d 16h

Table 4.6: Execution time for sRNARFTarget and IntaRNA for benchmarking data. Both programs were run on the same computer (see Table 4.9 for the computer specifications).

CopraRNA Webserver			
Bacteria	**sRNA**	**No. of homologs**	**Execution time**
Escherichia coli	arcZ	8	8h
Pasteurella multocida	GcvB	4	8h 19min
Synechocystis	IsrR1	19	17h 49min

Table 4.7: CopraRNA webserver job execution time

4.4 Interpreting sRNARFTarget predictions

We applied sRNARFTarget_SHAP and sRNARFTarget_CP to one sRNA-mRNA pair from each of the three bacteria that were predicted by sRNARFTarget with the highest prediction interaction probability. These three observations are true positives and were correctly classified by sRNARFTarget.

Figure 4.15 shows the plots generated by the sRNARFTarget_SHAP (using the SHAP package) program for *dsrA-hns* pair of *E. coli*. Figure 4.15(a) shows the SHAP's decision plot for this observation. We can see how the model has reached its decision from bottom to top. Coloured line is the observation. This plot shows how the model arrives at its decision using cumulative SHAP values. The X-axis is displaying the model's output (prediction probability). Y-axis is the features ordered by importance (shap values for features of the observation). SHAP values for each feature are added to the model's base value (average value of the model output over

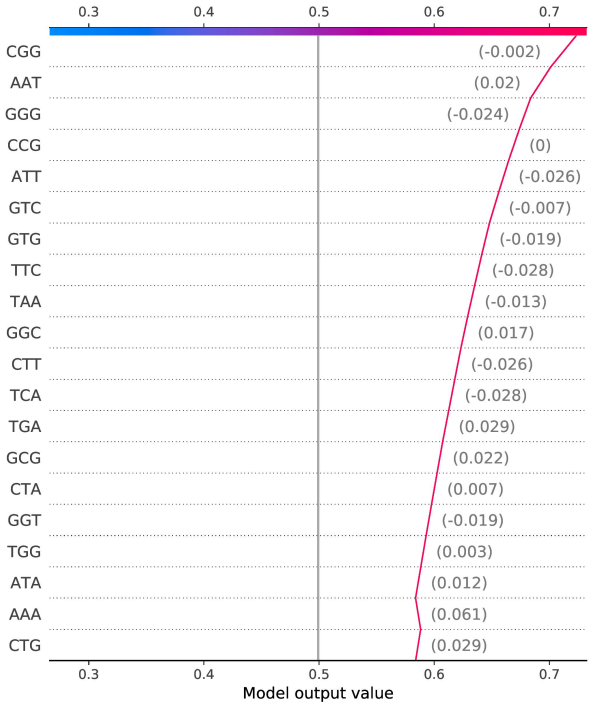
the training set) from bottom to top. Most of the features (of the top 20 significant features for this observation) except for feature AAA contribute to the predictions being positive.

Figure 4.15(b) is the waterfall plot generated by sRNARFTarget_SHAP for this observation; it is showing how the features sorted by their significance (shap values), move the model output from the base value. Less significant features are grouped at the bottom (if the number of features exceeds the max_display parameter).

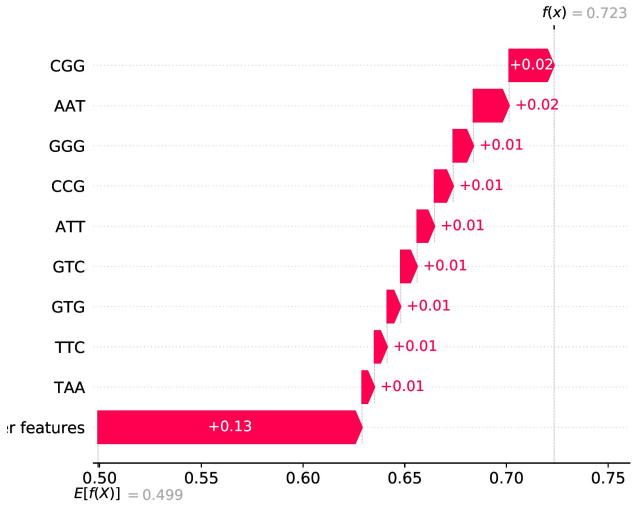
Figure 4.15(c) shows SHAP's force plot for this observation. This plot shows the output value of the prediction for the chosen observation and base value. Features pushing the prediction to be higher are in red, while those decreasing are in blue. Features CGG and AAT followed by GGG, CCG then ATT in red colour have the highest magnitude and are pushing the model prediction to be higher (more likely interaction) while feature AAA in blue is pushing it towards the negative side (less likely interaction).

Figure 4.16 (a) and (b) show the ceteris paribus plot generated from sRNARFTarget_CP (using pyCeterisparibus package) program for AAA and CGG features. Based on the visual results of sRNARFTarget_SHAP, we randomly selected two features AAA (decreasing the interaction prediction) and CGG (increasing the probability of interaction). We ran sRNARFTarget_CP for the same observation (*dsrA-hns* pair) to see the trend of these two features for *dsrA-hns* pair. This plot shows the real-time change in prediction for the selected variable as the feature value changes (Interactive

plot). The X-axis shows feature value; Y-axis shows model prediction.



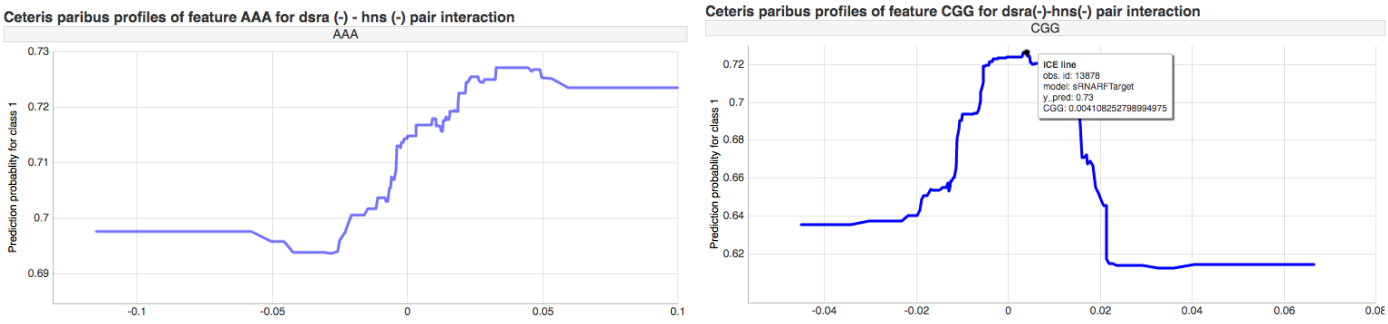
(a) Decision plot



(b) Waterfall plot

(c) Force plot

Figure 4.15: Decision, waterfall, and force plots for *E. coli dsrA-hns* pair's prediction made by sRNARFTarget.

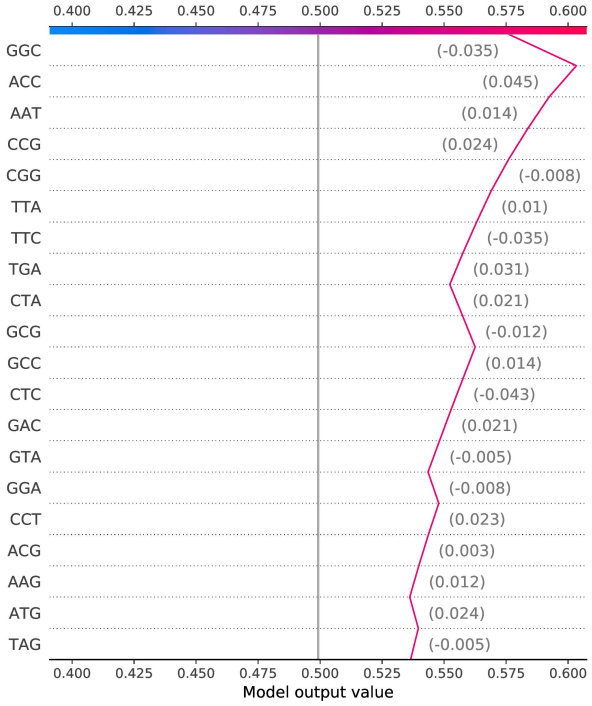


(a) (b)

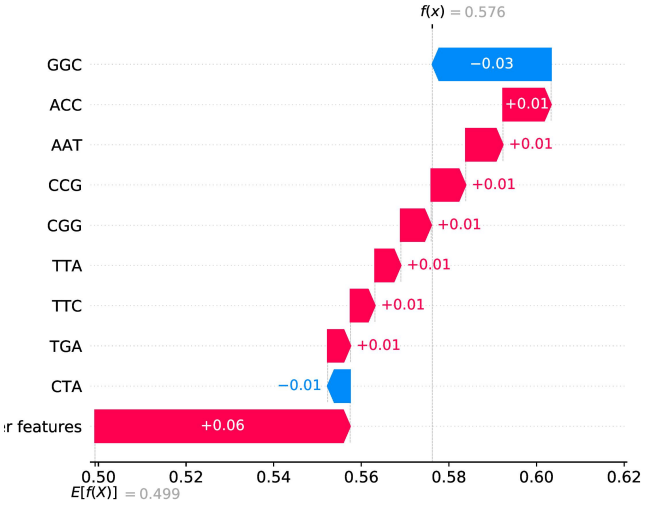
Figure 4.16: Ceteris paribus plot for features AAA and CGG for *E. coli dsrA-hns* pair's prediction made by sRNARFTarget.

Figure 4.17 (a), (b) and (c) plots are for *isaR1-petF* (ssl0020) pair from *Synechocystis*. The decision plot shows the way the model reached its decision. Waterfall plot displays that feature GGC is moving the prediction to lower and has the highest significance for this observation. Force plot shows that features ACC and AAT are pushing the prediction to be higher. Feature GGC in blue has the highest significance and is pushing the interaction prediction to be lower.

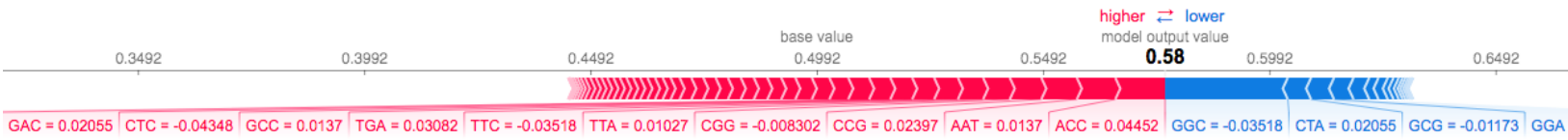
Figure 4.18 shows that ceteris paribus plot for feature GGC for *isaR1-petF* pair from *Synechocystis*. It shows the models' prediction for different values of GGC.



(a) Decision plot



(b) Waterfall plot



(c) Force plot

Figure 4.17: Decision, waterfall, and force plots for *Synechocystis isaR1-petF* pair's prediction made by sRNARFTarget.

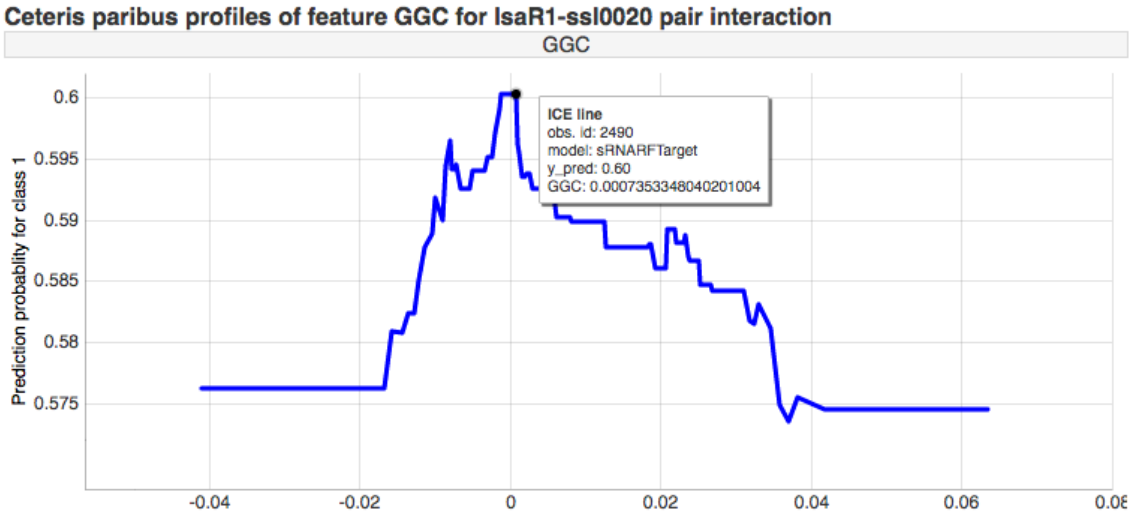
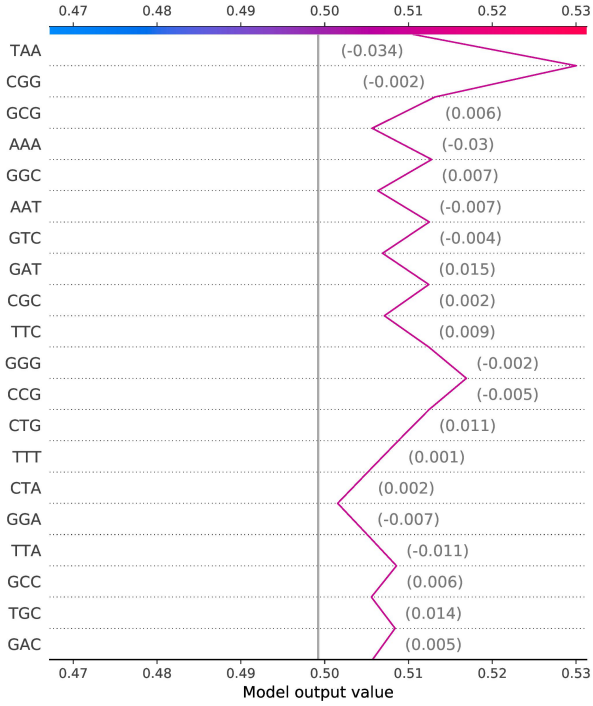
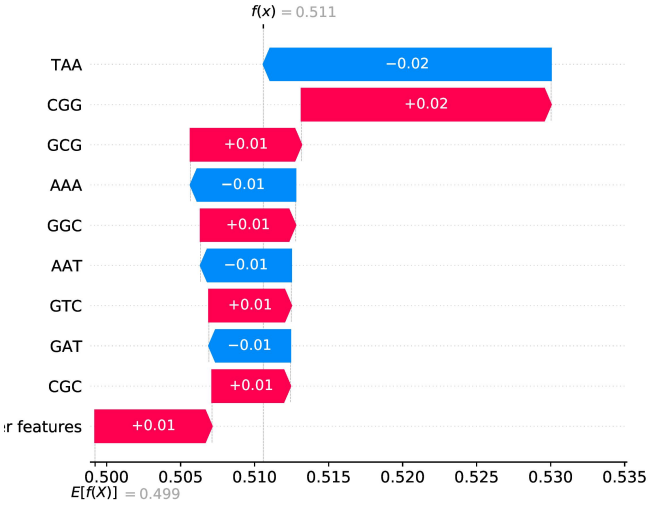


Figure 4.18: Ceteris paribus plot for feature GGC for *Synechocystis isaR1-petF* pair's prediction made by sRNARFTarget.

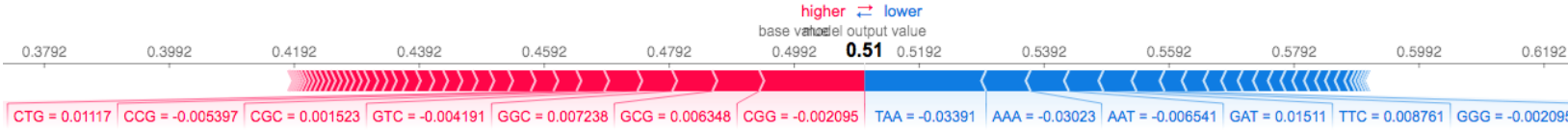
Figure 4.19 shows the SHAP plots for the interaction prediction made by sRNARF-Target for the *gcvB-metQ* pair of *P. multocida*. The model's decision path can be seen in the decision plot (a). Waterfall plot (b) shows that features TAA, AAA, AAT and GAT are moving the model's outcome to towards the base value (reducing the interaction probability) while features GCG, CGG, GGC, GTC and CGC are moving the model's outcome away from the base value (increase the chances of interaction). Force plot (c) is showing the model's outcome 51% interaction probability and how the features the pushing the prediction to be higher or lower. Features GCG, CGG, TAA and AAA seem to be the most significant features in predicting this observation.



(a) Decision plot



(b) Waterfall plot



(c) Force plot

Figure 4.19: Decision, waterfall, and force plots for *P. multocida gcvB-metQ* pair's prediction made by sRNARFTarget.

Figure 4.20 shows the ceteris paribus plot for *gcvB-metQ* pair of *P. multocida* and displays an interactive plot with different prediction outcomes for different values of

feature TAA.

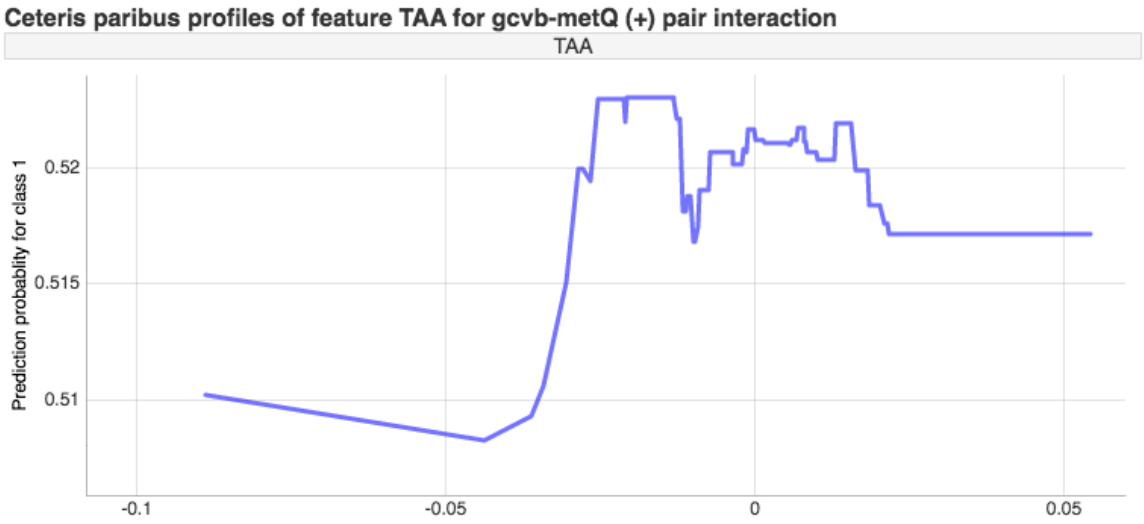


Figure 4.20: Ceteris paribus plot for feature TAA for *P. multocida gcvB-metQ* pair's prediction made by sRNARFTarget.

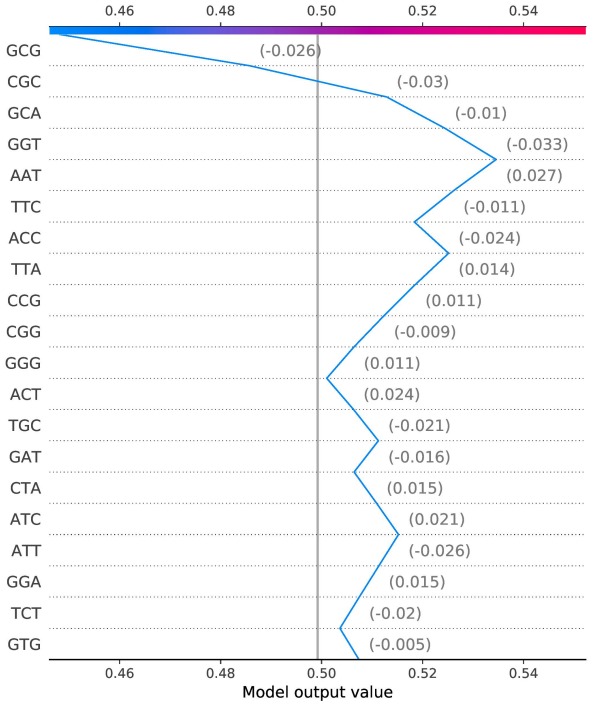
One of the use cases of interpretability programs can be when an instance gets misclassified. Here we discuss one such false-negative example. We take an observation that is a true interaction pair *omrA-ompT* of *E. coli* from the benchmarking set that was predicted to be non-interaction by sRNARFTarget with 0.45 predicted interaction probability.

Figure 4.21(a) shows the decision plot for this observation and how the model has made its decision. As this plot shows that features GCG and CGC along with other features are driving the models' outcome to be lower, we picked first two features based on decision plot GCG and CGC and plotted ceteris paribus plot for these two features shown in Figure 4.21 (c) and (d). We changed the value of CGC to -0.0119

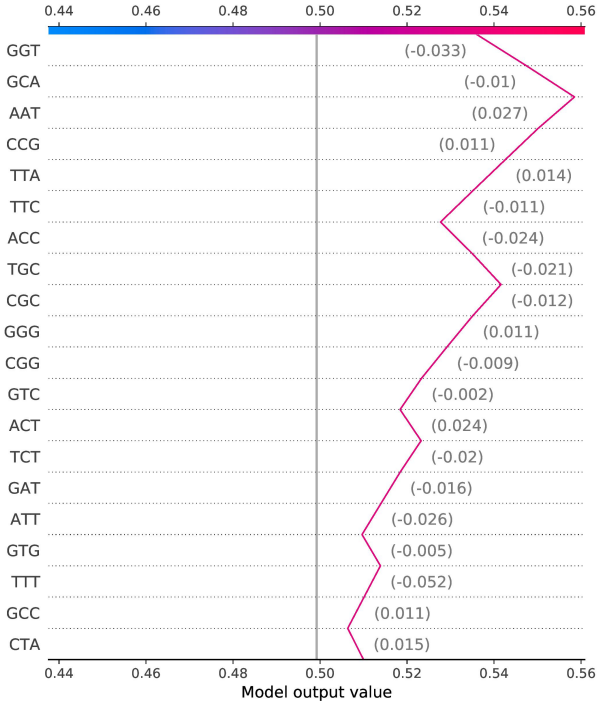
because the plots show that the lower negative value of CGC appears to increase the prediction probability of being this observation to have most likely interaction while the positive value pushes it to be non-interaction. Also, we changed the value of GCG to 0.0235 as the positive value near zero seems to increase prediction probability, while negative value decreases it.

After changing the values of GCG and CGC, we ran the sRNARFTarget_SHAP programs for this observation. Changing the feature values caused an increase in the prediction from 0.45 to 0.54 and the observation becomes correctly classified as shown in Figure 4.21(b). Program versions can be seen in Table 4.8. System specifications can be seen in Table 4.9.

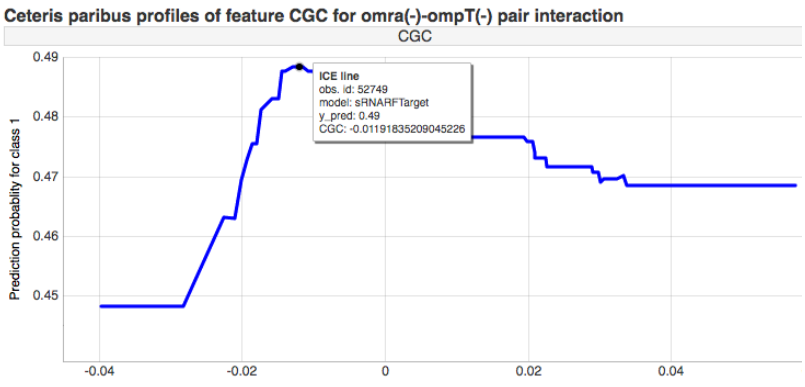
Table 4.10 shows the program execution times for sRNARFTarget_SHAP and sRNATarget_CP interpretability programs. sRNARFTarget_SHAP and sRNARFTarget_CP can be run for one sRNA-mRNA pair and one feature at a time respectively. sRNARFTarget_SHAP program took 5.35 seconds for one sRNA-mRNA pair. sRNARFTarget_CP program took 4.27 seconds for a single feature of one sRNA-mRNA pair.



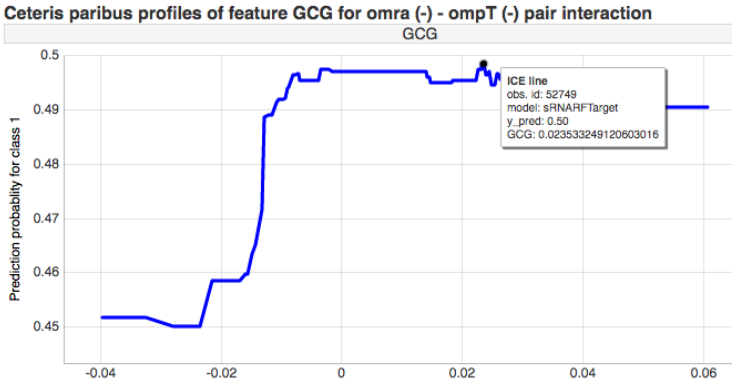
(a) Decision plot before changing values of CGC and GCG



(b) Decision plot after changing values of CGC and GCG



(c) Ceteris paribus plot for CGC



(d) Ceteris paribus plot for GCG

Figure 4.21: Missclassified *E. coli* instance

Software/Program	**Version**
Python	3.7.4
R	3.5.1
Nextflow	0.32.0
bedtools	2.27.1
CentroidFold	0.0.16
RNAdistance	2.4.13
Sklearn	0.22.1
skbio	0.5.5
pandas	0.25.1
numpy	1.18.1
SHAP	0.35.0
pyCeterisparibus	0.5.2
IntaRNA	3.1.0.2
Vienna RNA package	2.4.13
boost	1.70.0

Table 4.8: Software or program versions

Processor Name	Intel Core i7
Processor Speed	2.2 GHz
Number of Processors	1
Total Number of Cores	4
RAM	16 GB

Table 4.9: System specifications

Program	**Execution time**
sRNARFTarget_SHAP	5.35s
sRNARFTarget_CP	4.27s

Table 4.10: Execution time of sRNARFTarget interpretability programs

4.5 Summary

In this chapter, we present the results of assessing the performance of sRNARFTarget. A program for sRNA target prediction using a random forest model. We compared the performance in terms of AUROC and AUPRC of sRNARFTarget, CopraRNA and IntaRNA on independent datasets from three bacteria. Results show that CopraRNA performed best among all the three programs followed by sRNARFTarget then IntaRNA. However, CopraRNA is a comparative genomics-based approach that

needs sRNA and mRNAs to be conserved in at least four bacteria. In transcriptome-wide predictions, sRNARFTarget outperformed IntaRNA, a state-of-the-art non-comparative genomics approach, on all three bacteria. sRNARFTarget runs faster than CopraRNA and IntaRNA reducing the execution time to seconds instead of hours. As sRNARFTarget removes the conservation restriction of CopraRNA, we expect sRNARFTarget to be useful for predicting targets for sRNAs without homologs in other bacteria. As sRNARFTarget outperforms IntaRNA in terms of AUROC and execution time, sRNARFTarget should become the non-comparative genomics program to be used for sRNA target prediction. Additionally, sRNARFTarget is easy to use and provides interpretability functionality to facilitate the understanding of its predictions.

Chapter 5

Conclusion

This research focused on developing a transcriptome-wide sRNA target prediction program, sRNARFTarget. We collected experimentally verified sRNA-mRNA pairs from the literature to create a training data consisting of 745 interacting sRNA-mRNA pairs. We selected a random forest model as the final model for sRNARFTarget with the trinucleotide frequency difference between sRNA-mRNA as features. We validated sRNARFTarget in three very distinct bacterial species (*E.coli*, *Synechocystis*, and *P. multocida*). This suggests that the features employed capture a global interaction pattern shared by distinct bacterial species. sRNARFTarget uses the whole sequence for target prediction. To facilitate the use of sRNARFTarget, we created a nextflow pipeline. As future work, a web interface for sRNARFTarget can be developed.

In our benchmark, we compared sRNARFTarget with CopraRNA and IntaRNA. Our results show that the comparative genomics-based approach used by CopraRNA

is the best performing approach. For programs providing transcriptome-wide predictions (instead of predictions for a single sRNA such as CopraRNA), sRNARFTarget substantially outperforms IntaRNA in terms of AUROC, and rankings of true interacting pairs. Unlike CopraRNA, sRNARFTarget does not need an sRNA or mRNA sequence to be conserved among other bacteria and can generate predictions for any number of sRNA and mRNA sequences.

Additionally, we have facilitated the interpretation of predictions made by sRNARFTarget using existing python packages of interpretability. These interpretations visually show how the model reaches its decision. Our results show that the impact of sequence segments (trinucleotide frequency difference) differ from pair to pair. This can help understand microbiologists the precise sequence segments that contribute more to a specific sRNA-mRNA interaction. As IntraRNA and CopraRNA are not machine-learning based, interpretability programs cannot be directly applied to them. As future work, one could perform feature importance analysis in IntaRNA algorithm to compare with that of sRNARFTarget.

As CopraRNA performs best among all three programs, we suggest using CopraRNA when the homologs of the sRNA-mRNA sequences are available in at least four bacteria. For the transcriptome-wide prediction or when homolog sequences are not available, we recommend using sRNARFTarget as it has been shown to outperform IntaRNA.

Bibliography

[1] Adrien Pain, Alban Ott, Hamza Amine, Tatiana Rochat, Philippe Bouloc, and Daniel Gautheret. An assessment of bacterial small RNA target prediction programs. *RNA Biology*, 12(5):509–513, 2015. PMID: 25760244.

[2] E. Gerhart. Chapter Three - Small RNAs in Bacteria and Archaea: Who They Are, What They Do, and How They Do It. volume 90 of *Advances in Genetics*, pages 133 – 208. Academic Press, 2015.

[3] Patrick R. Wright, Andreas S. Richter, Kai Papenfort, Martin Mann, Jörg Vogel, Wolfgang R. Hess, Rolf Backofen, and Jens Georg. Comparative genomics boosts target prediction for bacterial small RNA. *Proceedings of the National Academy of Sciences*, 110(37):E3487–E3496, 2013.

[4] Andreas S. Richter, Anke Busch, and Rolf Backofen. IntaRNA: efficient prediction of bacterial sRNA targets incorporating target site accessibility and seed regions. *Bioinformatics*, 24(24):2849–2856, 10 2008.

[5] Alisa M. King, Carin K. Vanderpool, and Patrick H. Degnan. sRNA Target Prediction Organizing Tool (SPOT) Integrates Computational and Experimental Data To Facilitate Functional Characterization of Bacterial Small RNAs. *mSphere*, 4(1), 2019.

[6] Mary Beth Kery, Monica Feldman, Brian Tjaden, and Jonathan Livny. TargetRNA2: identifying targets of small regulatory RNAs in bacteria. *Nucleic Acids Research*, 42(W1):W124–W129, 04 2014.

[7] Brian Tjaden. TargetRNA: a tool for predicting targets of small RNA action in bacteria. *Nucleic Acids Research*, 36:W109–W113, 2008.

[8] Stephan H. Bernhart, Hakim Tafer, Ulrike Mukstein, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. Partition function and base pairing probabilities of RNA heterodimers. *Algorithms for Molecular Biology*, Mar 2006.

[9] Hofacker IL Tafer H. RNAplex: a fast tool for RNA–RNA interaction search. *Bioinformatics*, 24(22):2657–2663, 04 2008.

[10] Hackermüller J Bernhart SH Stadler PF Hofacker IL Mückstein U, Tafer H. Thermodynamics of RNA–RNA binding. *Bioinformatics*, 22(10):1177–1182, 01 2006.

[11] Andronescu Mirela, Zhang Zhi Chuan, and Condon Anne. Secondary Structure Prediction of Interacting RNA Molecules. *Journal of Molecular Biology*, 345(5):987 – 1001, 2005.

[12] Jan Krüger and Marc Rehmsmeier. RNAhybrid: microRNA target prediction easy, fast and flexible. *Nucleic Acids Research*, 34:W451–W454, 07 2006.

[13] I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie / Chemical Monthly*, 1994.

[14] Xiaomin Ying, Yuan Cao, Jiayao Wu, Qian Liu, Lei Cha, and Wuju Li. starpicker: A method for efficient prediction of bacterial srna targets based on a two-step model for hybridization. *PLOS ONE*, 6(7):1–12, 07 2011.

[15] Stadler PF Hofacker IL Eggenhofer F, Tafer H. RNApredator fast accessibility-based prediction of sRNA targets. *Nucleic Acids Research*, 39:W149–W154, 06 2011.

[16] Yong Zhang, Shiwei Sun, Tao Wu, Jie Wang, Changning Liu, Lan Chen, Xiaopeng Zhu, Yi Zhao, Zhihua Zhang, Baochen Shi, Hongchao Lu, and Runsheng Chen. Identifying Hfq-binding small RNA targets in *Escherichia coli*. *Biochemical and biophysical research communications*, 343(3):950—955, May 2006.

[17] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[18] Pierre Mandin, Francis Repoila, Massimo Vergassola, Thomas Geissmann, and Pascale Cossart. Identification of new noncoding RNAs in *Listeria monocyto-*

genes and prediction of mRNA targets. *Nucleic Acids Research*, 35(3):962–974, 01 2007.

[19] Mann, Martin and Wright, Patrick R. and Backofen, Rolf. IntaRNA 2.0: enhanced and customizable prediction of RNA–RNA interactions. *Nucleic Acids Research*, 45(W1):W435–W439, 05 2017.

[20] Athanasius F. Bompfünewerer, Rolf Backofen, Stephan H. Bernhart, Jana Hertel, Ivo L. Hofacker, Peter F. Stadler, and Sebastian Will. Variations on RNA folding and alignment: lessons from Benasque. *Journal of Mathematical Biology*, 56(1):129–144, 2008.

[21] Ross C Hardison. Comparative Genomics. *PLOS Biology*, 1(2), 11 2003.

[22] Rolf Backofen and Wolfgang R. Hess. Computational prediction of sRNAs and their targets in bacteria. *RNA Biology*, 7(1):33–42, 2010.

[23] Daniel Lai and Irmtraud M. Meyer. A comprehensive comparison of general RNA–RNA interaction prediction methods. *Nucleic Acids Research*, 44(7):e61–e61, 12 2015.

[24] Gissi C et al Mignone F. Untranslated regions of mRNAs. *Genome biology*, 2002.

[25] Zhong et al Wang. RNA-Seq: a revolutionary tool for transcriptomics. *Nature reviews. Genetics*, 10(1):57–63, 2009.

[26] David Lalaouna, Marie-Claude Carrier, Szabolcs Semsey, Jean-Simon Brouard, Jing Wang, Joseph T. Wade, and Eric Massé. A 3 External Transcribed Spacer in a tRNA Transcript Acts as a Sponge for Small RNAs to Prevent Transcriptional Noise. *Molecular Cell*, 58(3):393–405, 2015.

[27] Kook Han, Brian Tjaden, and Stephen Lory. GRIL-seq provides a method for identifying direct targets of bacterial small regulatory RNA by in vivo proximity ligation. *Nature Microbiology*, 2(3):16239, 2016.

[28] Shafagh A Waters, Sean P McAteer, Grzegorz Kudla, Ignatius Pang, Nandan P Deshpande, Timothy G Amos, Kai Wen Leong, Marc R Wilkins, Richard Strugnell, David L Gally, David Tollervey, and Jai J Tree. Small RNA interactome of pathogenic *E. coli* revealed through crosslinking of RNase E. *The EMBO Journal*, 36(3):374–387, 2017.

[29] Sahar Melamed, Asaf Peer, Raya Faigenbaum-Romm, Yair E. Gatt, Niv Reiss, Amir Bar, Yael Altuvia, Liron Argaman, and Hanah Margalit. Global Mapping of Small RNA-Target Interactions in Bacteria. *Molecular Cell*, 63(5):884 – 897, 2016.

[30] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational*

Linguistics: Demonstrations, pages 97–101. Association for Computational Linguistics, 2016.

[31] Brad Boehmke and Brandon M. Greenwell. Chapter 16 Interpretable Machine Learning — Hands-On Machine Learning with R. `https://bradleyboehmke.github.io/HOML/iml.html`, 2020.

[32] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics*, 8(8):832, 2019.

[33] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining Explanations: An Overview of Interpretability of Machine Learning. *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018.

[34] Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. Making machine learning models interpretable. In *ESANN*, volume 12, pages 163–172. Citeseer, 2012.

[35] Tae Keun Yoo, Ik Hee Ryu, Hannuy Choi, Jin Kuk Kim, In Sik Lee, Jung Sub Kim, Geunyoung Lee, and Tyler Hyungtaek Rim. Explainable Machine Learning Approach as a Tool to Understand Factors Used to Select the Refractive Surgery Technique on the Expert Level. *Translational Vision Science Technology*, 9(2), 2020.

[36] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[37] Gajendra Jung Katuwal and Robert Chen. Machine learning model interpretability for precision medicine. *arXiv preprint arXiv:1610.09045*, 2016.

[38] slundberg/shap: A game theoretic approach to explain the output of any machine learning model. `https://github.com/slundberg/shap`. (Accessed on 06/25/2020).

[39] Michał Kuźba, Ewa Baranowska, and Przemysław Biecek. pyceterisparibus: explaining machine learning models with ceteris paribus profiles in python. *Journal of Open Source Software*, 4(37):1389, 2019.

[40] 5.9 Shapley Values — Interpretable Machine Learning. `https://christophm.github.io/interpretable-ml-book/shapley.html#fn41`. (Accessed on 06/25/2020).

[41] Przemysław Biecek. GitHub - pbiecek/ceterisParibus: Ceteris Paribus Plots (What-If plots) for explanations of a single observation. `https://github.com/pbiecek/ceterisParibus`. (Accessed on 06/25/2020).

[42] Leo Breiman. Random Forests. *Machine Learning*, 45(1), 2001.

[43] Sahar Melamed, Asaf Peer, Raya Faigenbaum-Romm, Yair E. Gatt, Niv Reiss, Amir Bar, Yael Altuvia, Liron Argaman, and Hanah Margalit. Global Mapping of Small RNA-Target Interactions in Bacteria. *Molecular Cell*, 63(5):884 – 897, 2016.

[44] David Lalaouna, Audrey Morissette, Marie-Claude Carrier, and Eric Massé. DsrA regulatory RNA represses both hns and rbsD mRNAs through distinct mechanisms in *Escherichia coli*. *Molecular Microbiology*, 98(2):357–369, 2015.

[45] Mia K. Mihailovic, Jorge Vazquez-Anderson, Yan Li, Victoria Fry, Praveen Vimalathas, Daniel Herrera, Richard A. Lease, Warren B. Powell, and Lydia M. Contreras. High-throughput in vivo mapping of RNA accessible interfaces to identify functional sRNA binding sites. *Nature Communications*, 9(1):4084, 2018.

[46] Yi fan Zhang, Kook Sang Han, Courtney E. Chandler, Brian Tjaden, Robert K Ernst, and Stephen Lory. Probing the sRNA regulatory landscape of *P. aeruginosa*: post-transcriptional control of determinants of pathogenicity and antibiotic susceptibility. *Molecular microbiology*, 106 6:919–937, 2017.

[47] Tiago Pita, Joana Feliciano, and Jorge Leitão. Small Noncoding Regulatory RNAs from Pseudomonas aeruginosa and Burkholderia cepacia Complex. *International Journal of Molecular Sciences*, 19(12):3759, Nov 2018.

[48] Determination of the small RNA GcvB regulon in the Gram-negative bacterial pathogen *Pasteurella multocida* and identification of the GcvB seed binding re-

gion. *Rna-A Publication of the Rna Society*, 24, 2018.

[49] Kathrin S. Fröhlich, Katharina Haneke, Kai Papenfort, and Jörg Vogel. The target spectrum of SdsR small RNA in Salmonella. *Nucleic Acids Research*, 44(21):10406–10422, 2016.

[50] Daniel Ryan, Mohana Mukherjee, and Mrutyunjay Suar. The expanding targetome of small RNAs in *Salmonella Typhimurium*. *Biochimie*, 137:69 – 77, 2017.

[51] Juntao Mai, Chitong Rao, Jacqueline Watt, Xian Sun, Chen Lin, Lu Zhang, and Jun Liu. *Mycobacterium tuberculosis* 6C sRNA binds multiple mRNA targets via C-rich loops independent of RNA chaperones. *Nucleic Acids Research*, 47(8):4292–4307, 2019.

[52] Jens Georg, Gergana Kostova, Linda Vuorijoki, Verena Schön, Taro Kadowaki, Tuomas Huokko, Desirée Baumgartner, Maximilian Müller, Stephan Klähn, Yagut Allahverdiyeva, Yukako Hihara, Matthias E. Futschik, Eva-Mari Aro, and Wolfgang R. Hess. Acclimation of Oxygenic Photosynthesis to Iron Starvation Is Controlled by the sRNA IsaR1. *Current Biology*, 27(10):1425–1436.e7, 2017.

[53] Jens Georg, Dennis Dienst, Nils Schürgers, Thomas Wallner, Dominik Kopp, Damir Stazic, Ekaterina Kuchmina, Stephan Klähn, Heiko Lokstein, Wolfgang R. Hess, and Annegret Wilde. The Small Regulatory RNA SyR1/PsrR1 Controls Photosynthetic Functions in Cyanobacteria. *The Plant Cell*, 26(9):3661–3679, 2014.

[54] Sabine Brantl and Reinhold Brückner. Small regulatory RNAs from low-GC Gram-positive bacteria. *RNA Biology*, 11(5):443–456, 2014. PMID: 24576839.

[55] Wang, Jiang and Liu, Tao and Zhao, Bo and Lu, Qixuan and Wang, Zheng and Cao, Yuan and Li, Wuju. sRNATarBase 3.0: an updated database for sRNA-target interactions in bacteria. *Nucleic Acids Research*, 44(D1):D248–D253, 10 2015.

[56] Mario Tello, Felipe Avalos, and Omar Orellana. Codon usage and modular interactions between messenger RNA coding regions and small RNAs in *Escherichia coli*. *BMC Genomics*, 19(1):657, 2018.

[57] Paolo Di Tommaso, Maria Chatzou, Evan W. Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35, Apr 2017.

[58] Eric Sayers. *Entrez Programming Utilities Help [Internet]*. 2008.

[59] Aaron R. Quinlan and Ira M. Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 01 2010.

[60] biocontainers/bedtools - Docker Hub. `https://hub.docker.com/r/biocontainers/bedtools/`. (Accessed on 06/10/2020).

[61] Sequences (skbio.sequence) — scikit-bio 0.5.6 documentation. `http://scikit-bio.org/docs/latest/sequence.html`. (Accessed on 06/04/2020).

[62] Michiaki Hamada, Hisanori Kiryu, Kengo Sato, Toutai Mituyama, and Kiyoshi Asai. Prediction of RNA secondary structure using generalized centroid estimators. *Bioinformatics*, 25(4):465–473, 2008.

[63] Ronny Lorenz, Stephan H. Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[65] importance function — R Documentation. `https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/importance`. (Accessed on 06/04/2020).

[66] Receiver Operating Characteristic (ROC) with cross validation — scikit-learn 0.23.1 documentation. `https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html#sphx-glr-auto-examples-model-selection-plot-roc-crossval-py`. (Accessed on 06/10/2020).

[67] Precision-Recall — scikit-learn 0.23.1 documentation. `https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.`

html. (Accessed on 06/10/2020).

[68] pandas.DataFrame.subtract — pandas 1.0.5 documentation. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.subtract.html. (Accessed on 06/23/2020).

[69] Amazon Web Services - Documentation. https://github.com/awsdocs. (Accessed on 06/24/2020).

[70] phdegnan/SPOT: sRNA-target Prediction Organizing Tool. https://github.com/phdegnan/SPOT. (Accessed on 06/24/2020).

[71] Steffen C. Lott, Richard A Schäfer, Martin Mann, Rolf Backofen, Wolfgang R Hess, Björn Voss, and Jens Georg. GLASSgo - Automated and reliable detection of sRNA homologs from a single input sequences. *Frontiers in Genetics*, 9:124, 2018.

[72] Backofenlab/intarna: Efficient target prediction incorporating accessibility of interaction sites. https://github.com/BackofenLab/IntaRNA/#install. (Accessed on 07/22/2020).

[73] IntaRNA/R at master · BackofenLab/IntaRNA. https://github.com/BackofenLab/IntaRNA/tree/master/R#intarna_csv_p-valuer. (Accessed on 06/10/2020).

[74] Jan Grau, Ivo Grosse, and Jens Keilwagen. PRROC: computing and visualizing precision-recall and receiver operating characteristic curves in R. *Bioinformatics*, 31(15):2595–2597, 2015.

[75] ggplot2 violin plot : Quick start guide - R software and data visualization - Easy Guides - Wiki - STHDA. `http://www.sthda.com/english/wiki/ggplot2-violin-plot-quick-start-guide-r-software-and-data-visualization`. (Accessed on 06/10/2020).

[76] Ranking — Ranking 0.3.1 documentation. `https://pythonhosted.org/ranking/`. (Accessed on 06/10/2020).

[77] Function reference • ggplot2. `https://ggplot2.tidyverse.org/reference/`. (Accessed on 06/10/2020).

[78] Winston Haynes. *Wilcoxon Rank Sum Test*, pages 2354–2355. Springer New York, New York, NY, 2013.

[79] wilcox.test function — r documentation. `https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/wilcox.test`. (Accessed on 07/10/2020).

[80] Yanzhen Xu, Xiaohan Zhao, Shuai Liu, Shichao Liu, Yanqing Niu, Wen Zhang, and Leyi Wei. LncPred-IEL: A Long Non-coding RNA Prediction Method using Iterative Ensemble Learning. *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2019.

[81] Aimin Li, Junying Zhang, and Zhongyin Zhou. PLEK: a tool for predicting long non-coding RNAs and messenger RNAs based on an improved k-mer scheme. *BMC Bioinformatics*, 15(1):311, 2014.

[82] Dau Phan, Ngoc Giang Nguyen, Favorisen Rosyking Lumbanraja, Mohammad Reza Faisal, Bahriddin Abapihi, Bedy Purnama, Mera Kartika Delimayanti, Mamoru Kubo, and Kenji Satou. Combined Use of k-mer Numerical Features and Position-Specific Categorical Features in Fixed-Length DNA Sequence Classification. *Journal of Biomedical Science and Engineering*, 2017.

[83] Tzu-Hao Chang, Li-Ching Wu, Jun-Hong Lin, Hsien-Da Huang, Baw-Jhiune Liu, Kuang-Fu Cheng, and Jorng-Tzong Horng. Prediction of small non-coding RNA in bacterial genomes using support vector machines. *Expert Systems with Applications*, 37(8):5549 – 5557, 2010.

[84] Gisela Storz, Jörg Vogel, and Karen M. Wassarman. Regulation by Small RNAs in Bacteria: Expanding Frontiers. *Molecular Cell*, 43(6):880–891, 2011.

[85] Markus Fricke, Ruman Gerst, Bashar Ibrahim, Michael Niepmann, and Manja Marz. Global importance of RNA secondary structures in protein-coding sequences. *Bioinformatics*, 35(4):579–583, 2018.

[86] Joris Sansen, Patricia Thebault, Isabelle Dutour, and Romain Bourqui. Visualization of sRNA-mRNA Interaction Predictions. *2016 20th International Conference Information Visualisation (IV)*, 2016.

[87] Zuzanna Wroblewska and Mikolaj Olejniczak. Hfq assists small RNAs in binding to the coding sequence of ompD mRNA and in rearranging its structure, 2016.